UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Desenvolvimento de Modelos de Reconhecimento Facial Baseados em Inteligência Computacional para Mitigar Viés e Promover Equidade

Marcello Vinicius Alves Ozzetti Cruz

Monografia - MBA em Inteligência Artificial e Big Data

ICMC USP SÃO CARLOS

Marcello Vinicius Alves Ozzetti Cruz

Desenvolvimento de Modelos de Reconhecimento Facial Baseados em Inteligência Computacional para Mitigar Viés e Promover Equidade

Monografia apresentada ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador: Prof. PhD. Jó Ueyama

Versão original

São Paulo

2024

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi, ICMC/USP, com os dados fornecidos pelo(a) autor(a)

S856m

Ozzetti, Marcello
Desenvolvimento de Modelos de Reconhecimento Facial Baseados em Inteligência Computacional para Mitigar Viés e Promover Equidade / Marcello Vinicius Alves Ozzetti Cruz ; orientador Jó Ueyama. – São Paulo, 2024.
95 p. : il. (algumas color.) ; 30 cm.

Monografia (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2024.

1. LaTeX. 2. abnTeX. 3. Classe USPSC. 4. Editoração de texto. 5. Normalização da documentação. 6. Tese. 7. Dissertação. 8. Documentos (elaboração). 9. Documentos eletrônicos. I. Ueyama, Jó, orient. II. Título.

Marcello Vinicius Alves Ozzetti Cruz

Developing Computational Intelligence-Based Facial Recognition Models to Mitigate Bias and Promote Equity

Monograph presented to the Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, as part of the requirements for obtaining the title of Specialist in Artificial Intelligence and Big Data.

Concentration area: Artificial Intelligence

Original version

São Paulo

2024

Este trabalho é dedicado aos alunos da USP, como uma contribuição das Bibliotecas do Campus USP de São Carlos para o desenvolvimento e disseminação da pesquisa científica da Universidade.

AGRADECIMENTOS

Agradeço a todos que de maneira direta, ou indireta me apoiaram na elaboração deste trabalho.

"O estudo, a busca da verdade e da beleza são domínios em que nos é consentido sermos crianças por toda a vida."

Albert Einstein

RESUMO

OZZETTI, Marcello **Desenvolvimento de Modelos de Reconhecimento Facial Baseados em Inteligência Computacional para Mitigar Viés e Promover Equidade**. 2024. 95 p. Monografia (MBA em Inteligência Artificial e Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Paulo, 2024.

A rápida evolução da Inteligência Artificial (IA) e o crescente uso de sistemas de reconhecimento faciais têm gerado desafios significativos, especialmente em relação aos preconceitos algorítmicos que afetam a precisão e a equidade desses sistemas. Esta pesquisa abordou as principais técnicas utilizadas no reconhecimento de biometria facial, focando especialmente em redes neurais convolucionais (CNNs) e suas aplicações práticas. Foram discutidos aspectos fundamentais como o pré-processamento de imagens, a extração de características e o uso de arquiteturas avançadas, incluindo ArcFace e ResNet50. O objetivo foi testar diferentes combinações de funções de perda e otimizadores para avaliar o impacto nas métricas, principalmente a acurácia, levando em consideração também o *scheduler* de aprendizado. O uso de técnicas de *data augmentation* e o uso de conjuntos de dados balanceados também foram aplicados.

Através da Arquitetura CNN LResNet50E-IR, foram realizados experimentos divididos em diferentes configurações, avaliando as funções de perda CrossEntropyLoss e ArcFaceLoss em combinação com os otimizadores SGD e AdamW. Os resultados revelaram que a combinação do otimizador AdamW com a função CrossEntropyLoss levou a um aumento significativo na acurácia, em comparação com outras configurações, atingindo variações de 0.58 a 0.95. Essas descobertas reforçam a importância da escolha adequada de funções de perda e otimizadores na construção de modelos eficazes para reconhecimento facial.

Os resultados obtidos desta pesquisa não apenas contribuem para o avanço do conhecimento acadêmico, mas também têm implicações práticas para a indústria. As diretrizes e melhores práticas identificadas podem ser aplicadas no aprimoramento da eficiência e precisão dos sistemas de reconhecimento facial de aplicações comerciais, segurança e de saúde. O desenvolvimento de soluções justas e inclusivas é crucial para garantir que os benefícios da tecnologia sejam acessíveis a todos.

Por fim, este trabalho oferece uma visão ampla sobre os desafios e oportunidades no campo do reconhecimento facial, contribuindo para um paradigma mais ético e inclusivo na aplicação da Inteligência Artificial (IA).

Palavras-chave: Inteligência Artificial. Redes Neurais. Viés em Biometria Facial.

ABSTRACT

OZZETTI, Marcello **Developing Computational Intelligence-Based Facial Recognition Models to Mitigate Bias and Promote Equity**. 2024. 95 p. Monograph (MBA in Artificial Intelligence and Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Paulo, 2024.

The rapid evolution of Artificial Intelligence (AI) and the growing use of facial recognition systems have generated significant challenges, especially concerning algorithmic biases that impact the accuracy and fairness of these systems. This research explored key techniques used in facial biometrics recognition, with a particular focus on convolutional neural networks (CNNs) and their practical applications. Fundamental aspects were discussed, such as image preprocessing, feature extraction, and the use of advanced architectures, including ArcFace and ResNet50. The aim was to test different combinations of loss functions and optimizers to assess the impact on metrics, especially accuracy, while also considering the learning scheduler. Data augmentation techniques and balanced datasets were also applied.

Through the CNN architecture LResNet50E-IR, experiments were conducted across different configurations, evaluating the loss functions CrossEntropyLoss and ArcFaceLoss in combination with the optimizers SGD and AdamW. The results revealed that the combination of the AdamW optimizer with the CrossEntropyLoss function led to a significant increase in accuracy compared to other configurations, with variations reaching from 0.58 to 0.95. These findings underscore the importance of choosing appropriate loss functions and optimizers in building effective facial recognition models.

The results obtained from this research not only contribute to the advancement of academic knowledge but also have practical implications for the industry. The guidelines and best practices identified can be applied to enhance the efficiency and accuracy of facial recognition systems in commercial, security, and healthcare applications. Developing fair and inclusive solutions is crucial to ensuring that the benefits of technology are accessible to everyone.

Finally, this work provides a broad perspective on the challenges and opportunities in the field of facial recognition, contributing to a more ethical and inclusive paradigm in the application of Artificial Intelligence (AI).

Keywords: Artificial Intelligence. Neural Network. Bias in Facial Biometrics.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

USP	Universidade de São Paulo
USPSC	Campus USP de São Carlos
RNA	Rede Neural Artificial
IA	Inteligencia Artificial
MLP	Rede Perceptron Multicamadas
CNN	Rede Neural Convolucional
RNN	Rede Neural Recorrente
GAN	Rede Neural Generativa Adversária
KDD	*Knowledge Discovery in Databases*
CPU	Unidade Central de Processamento
GPU	Unidade de Processamento Gráfico
GD	Gradiente Descendente
SGD	*Stochastic Gradient Descent*
TP	*True Positives*
FP	*False Positives*
2D	Duas Dimensões
3D	Três Dimensões
RMSprop	*Root Mean Square Propagation*
LBPH	*Local Binary Patterns Histogram*
MTCNN	*Multi-task Cascaded Convolutional Networks*
OpenCV	*Open Source Computer Vision Library*
PCA	Análise de Componentes Principais
LDA	Análise Discriminante Linear

SUMÁRIO

1 INTRODUÇÃO

A rápida evolução da Inteligência Artificial (IA) e o crescente papel dos sistemas de reconhecimento de imagem têm introduzido desafios significativos, destacando a necessidade urgente de abordar os preconceitos inerentes a essas tecnologias. O reconhecimento facial possui uma vasta gama de aplicações, desde segurança e verificação de identidade até comunicação online, transações bancárias e entretenimento digital. Embora a pesquisa em reconhecimento facial tenha começado na década de 1960, como apontam Stan Z., Li e Jain (Li; Jain, 2011), essa área ainda enfrenta desafios não resolvidos. Recentemente, os avanços em modelagem e técnicas de análise facial têm impulsionado o progresso, mas o reconhecimento facial confiável continua a ser um desafio para pesquisadores de visão computacional e reconhecimento de padrões. Drozdowski (Drozdowski *et al.*, 2020) ressalta que "existem inúmeras preocupações quanto à precisão e à justiça dos sistemas automatizados de tomada de decisão".

Entre os grupos étnicos frequentemente afetados por vieses algorítmicos, as pessoas negras são particularmente impactadas. Buolamwini e Gebru (Buolamwini; Gebru, 2018) destacam "disparidades substanciais na precisão da classificação de mulheres de pele mais escura, mulheres de pele mais clara, homens de pele mais escura e homens de pele mais clara em sistemas de classificação de gênero", sublinhando a necessidade urgente de que empresas comerciais construam algoritmos de análise facial que sejam verdadeiramente justos, transparentes e responsáveis. Além disso, a sub-representação de comunidades negras nos conjuntos de dados usados para treinar algoritmos perpetua desigualdades sistêmicas, conforme destacado em estudos como o de Martin (Martin, 2022).

Outro desafio no reconhecimento facial relaciona-se ao uso de tecnologias para auxiliar pessoas com deficiência visual, especialmente em validações biométricas. Modelos de reconhecimento facial prometem ajudar essas pessoas a realizarem transações e identificações utilizando biometria facial, conforme explorado no estudo de Jafri e Arabnia (Jafri; Ali; Arabnia, 2013), que menciona uma variedade de softwares, mecanismos e artigos relacionados.

As falhas nos modelos de reconhecimento facial frequentemente decorrem de dados de treinamento distorcidos, incompletos, desatualizados, desproporcionais ou que carregam preconceitos históricos, o que compromete o treinamento do algoritmo e perpetua esses preconceitos.

Esta dissertação propõe abordar essas questões, concentrando-se na construção de modelos de redes neurais para reconhecimento facial que visam mitigar o preconceito contra pessoas negras e evitar falhas e vieses em indivíduos com deficiência visual. O

desenvolvimento desses modelos não só pretende aprimorar a precisão dos sistemas, mas também garantir equidade, transparência e justiça ao enfrentar os preconceitos embutidos, beneficiando diversas aplicações do reconhecimento de imagem, como o aumento da acurácia em transações bancárias que requerem identificação biométrica. Ao fazê-lo, esta pesquisa busca contribuir para um paradigma mais ético e inclusivo na aplicação da Inteligência Artificial (IA), reconhecendo a importância de soluções tecnológicas que respeitem e promovam a inclusão humana (Diakopoulos, 2016).

A pesquisa adotará uma abordagem experimental para desenvolver e avaliar modelos de redes neurais que buscam mitigar vieses no reconhecimento facial. O processo incluirá:

1. Coleta e preparação de um conjunto de dados diversificado, com ênfase na representatividade de diferentes grupos étnicos e pessoas com deficiência visual.

2. Treinamento de modelos utilizando técnicas de aprendizado profundo, como redes neurais convolucionais e arquiteturas específicas para mitigação de preconceitos.

3. Avaliação dos modelos em termos de precisão, equidade e justiça, utilizando métricas como taxa de erro e análises de impacto de viés.

O trabalho utilizará as Redes Neurais Convolucionais (CNNs), abrangendo uma série de técnicas e arquiteturas avançadas que busquem a eficiência na análise de imagens. Inicialmente, o pré-processamento de imagens deverá incluir a normalização e padronização dos dados para garantir que as entradas estejam em uma escala adequada para o treinamento do modelo, além de técnicas de *data augmentation* para expandir artificialmente o conjunto de dados e prevenir *overfitting*. A extração de características será fundamental para identificar padrões visuais, bordas e texturas que são essenciais para a classificação e reconhecimento. Arquiteturas avançadas como LeNet, AlexNet, VGGNet e ResNet demonstram serem cruciais na evolução das CNNs, com inovações que melhoraram significativamente a precisão e a capacidade de generalização dos modelos.

No contexto do reconhecimento de biometria facial, será feita uma revisão das técnicas clássicas como Eigenfaces, Fisherfaces e LBPH, bem como os modelos avançados com FaceNet, DeepFace, VGG-Face e ArcFace, destacados por suas abordagens inovadoras, como o uso de perdas angulares e técnicas de alinhamento facial para melhorar a discriminação e a precisão do reconhecimento facial. A implementação prática do reconhecimento facial envolverá um pipeline que inclui detecção, alinhamento e extração de características, utilizando ferramentas e *frameworks* como PyTorch, OpenCV, TensorFlow e Keras. Esses recursos visão proporcionar flexibilidade e eficiência na construção de sistemas de reconhecimento facial robustos e escaláveis.

Esta dissertação está estruturada:

- **Introdução**: Contextualização do problema, objetivos e justificativa da pesquisa.
- **Fundamentação Teórica**: Síntese dos estudos existentes sobre reconhecimento facial, vieses algorítmicos e inclusão de pessoas com deficiência visual. Também são exploradas as Redes Neurais Artificiais aplicadas ao reconhecimento biométrico.
- **Metodologia**: Descrição detalhada das etapas de desenvolvimento e avaliação dos modelos propostos.
- **Avaliação Experimental**: Apresentação dos modelos construídos, resultados das avaliações e discussão dos achados.
- **Conclusão**: Resumo dos principais resultados, contribuições da pesquisa e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Para entender o funcionamento das Redes Neurais Artificiais (RNAs), é essencial compreender os conceitos básicos que relacionam o funcionamento do cérebro humano e seus componentes, os neurônios. Neste capítulo, será realizada uma revisão sobre a formação das conexões entre as células nervosas e as considerações sobre o modelo matemático que serve de base para a aprendizagem de máquina e para as redes neurais.

Além disso, serão abordados os conceitos fundamentais das RNAs, incluindo suas definições e diferentes arquiteturas. Também exploraremos as Redes Neurais Convolucionais (CNNs) e seu papel crucial no reconhecimento e processamento de imagens, com foco específico na identificação biométrica.

Para concluir o capítulo, analisaremos as tendências atuais, as ferramentas disponíveis e as arquiteturas avançadas de redes neurais que estão moldando o futuro da inteligência artificial.

2.1 Redes Neurais Artificiais

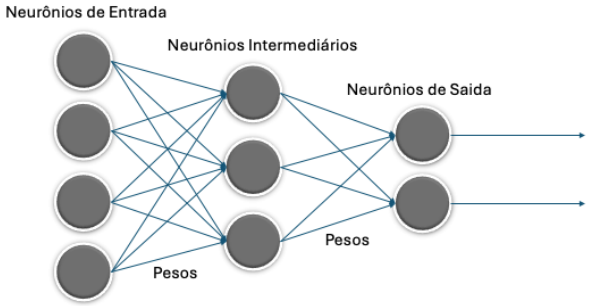
2.1.1 Definição

As Redes Neurais Artificiais (RNAs) são modelos computacionais inspirados no funcionamento do cérebro humano, simulando uma rede de neurônios conectados e organizados em camadas. Cada camada processa a informação recebida e a transmite para a próxima camada (Haykin, 2009). Embora não existam evidências científicas de que o cérebro humano opere exatamente como os mecanismos de aprendizagem usados em RNAs (Chollet, 2017), o estudo da Aprendizagem Profunda baseia-se em modelos e estruturas matemáticas que permitem compreender e reproduzir processos de aprendizagem.

A Aprendizagem Profunda (*Deep Learning*) é uma subárea do aprendizado de máquina que utiliza múltiplas camadas consecutivas para melhorar a capacidade de aprendizado e processamento de dados. O termo "aprendizado profundo"refere-se ao número de camadas em um modelo. Em geral, essas camadas são compostas por Redes Neurais Artificiais (Chollet, 2017).

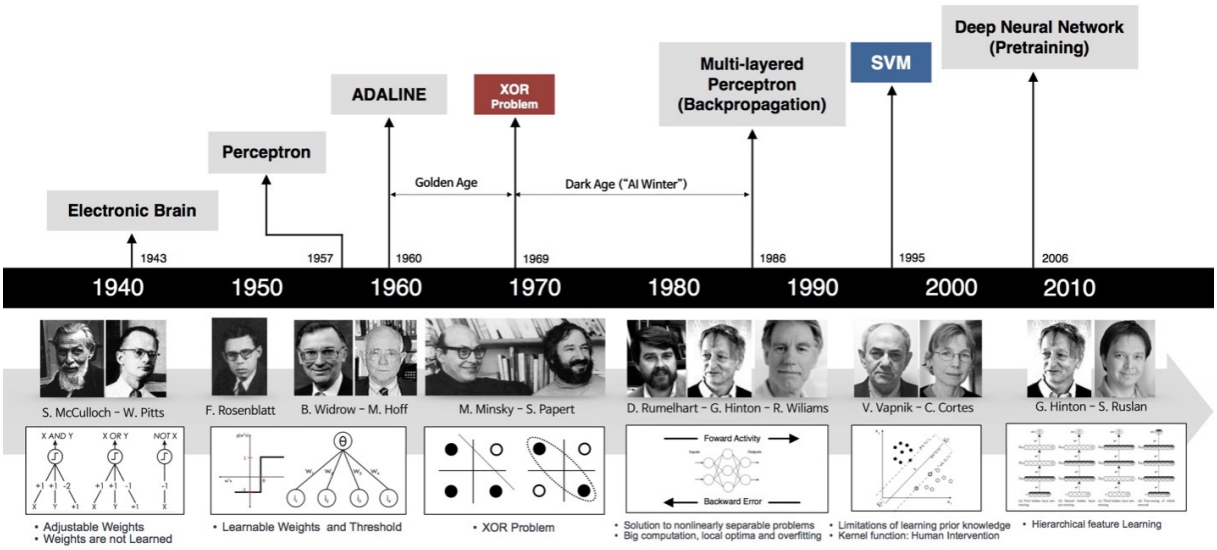
Uma RNA é composta por camadas encadeadas que mapeiam os dados para realizar predições com base em classes estabelecidas. Cada camada de entrada contém neurônios que codificam os valores de entrada e os transmitem como saída para as camadas subsequentes. As camadas possuem parâmetros a serem estimados, conhecidos como pesos, que armazenam o conhecimento adquirido durante o treinamento. Esses pesos são ajustados conforme as camadas extraem informações dos dados de entrada. Assim, a rede aprende com os dados e consegue estimar uma saída, conforme ilustrado na Figura 1.

Figura 1 – Exemplo de uma RNA com 2 Camadas e 4 Entradas com 2 Saídas



Fonte: Próprio Autor

Figura 2 – Marcos no Desenvolvimento das Redes Neurais



Fonte: (Academy, 2018)

2.1.2 Breve Histórico do Desenvolvimento das RNAs

Para compreender o estado atual das RNAs, é fundamental conhecer a trajetória de sua evolução. A Figura 2 resume alguns marcos na pesquisa e desenvolvimento das Redes Neurais Artificiais, conforme explorado no livro da Academia de Aprendizagem Profunda (Academy, 2018).

- **1943**: Warren McCulloch e Walter Pitts (Mcculloch; Pitts, 1943) criam um modelo computacional de redes neurais baseado em algoritmos de lógica de limiar.
- **1957**: Frank Rosenblatt (Rosenblatt, 1957) desenvolve o Perceptron, um algoritmo de reconhecimento de padrões baseado em uma rede neural de duas camadas usando operações de adição e subtração simples.
- **1980**: Kunihiko Fukushima (Fukushima, 1980) propõe a Rede Neocognitron, uma rede neural hierárquica e multicamada utilizada para reconhecimento de caligrafia e

outros problemas de reconhecimento de padrões.

- **1989**: Cientistas desenvolvem os primeiros algoritmos que utilizam redes neurais profundas, embora ainda com tempos de treinamento elevados.
- **1992**: Juyang Weng e Huang publicam o Cresceptron (Weng; Ahuja; Huang, 1992), um método para reconhecimento automático de objetos 3D a partir de dados desordenados.
- **2006**: O termo Aprendizagem Profunda (*Deep Learning*) ganha popularidade após Geoffrey Hinton e Ruslan Salakhutdinov (Hinton; Salakhutdinov, 2006) demonstrarem como uma rede neural de múltiplas camadas pode ser treinada de forma eficiente.
- **2009**: O NIPS Workshop sobre Aprendizagem Profunda para reconhecimento de voz apresenta técnicas de aprendizado profundo que não requerem pré-treinamento.
- **2012**: Algoritmos de reconhecimento de padrões artificiais alcançam desempenho comparável ao humano em tarefas simples.
- **2015**: O Facebook começa a utilizar Aprendizagem Profunda para identificar automaticamente usuários em fotografias.
- **2016**: O algoritmo *AlphaGo* da Google derrota o campeão mundial de Go, Lee Sedol, em um torneio na Coreia do Sul.
- **2017**: Empresas adotam Aprendizagem Profunda em diversas aplicações, impulsionando pesquisas e tecnologias ligadas a *Data Science*, Inteligência Artificial e *Big Data*.

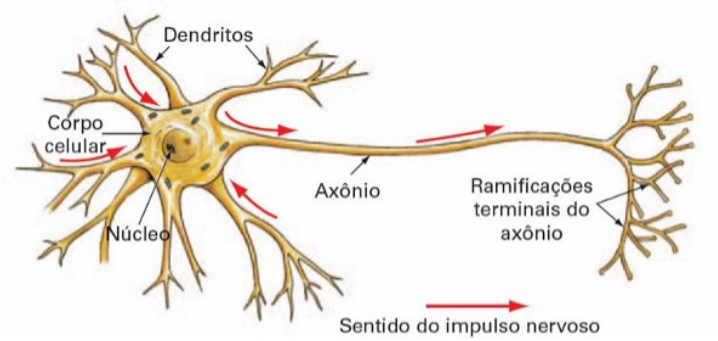
A partir da década de 1980, houve uma revolução nos estudos sobre redes neurais, tanto pelas características dos modelos propostos quanto pelas condições tecnológicas que possibilitaram o desenvolvimento de arquiteturas neurais mais robustas e o uso de *hardwares* mais avançados. As redes neurais profundas, também conhecidas como Aprendizagem Profunda (*Deep Learning*), emergiram como uma evolução natural das redes neurais.

2.1.3 O Neurônio Biológico e o Neurônio Artificial

Tanto no livro de Haykin (Haykin, 2001) quanto na obra da Academia de Aprendizagem Profunda (Academy, 2018), são apresentados os conceitos de neurônio biológico e neurônio artificial, resumidos a seguir.

O neurônio biológico é a unidade básica do cérebro humano, responsável pela transmissão de informações. O cérebro é composto por bilhões de neurônios interconectados, formando uma vasta rede de comunicação — a rede neural. Cada neurônio possui um corpo celular, diversos dendritos e um axônio. Os dendritos recebem sinais elétricos de outros

Figura 3 – Representação Simplificada do Neurônio Biológico



Fonte: (Academy, 2018)

neurônios através das sinapses, que são processados pelo corpo celular e transmitidos a outros neurônios. Os sinais transmitidos são impulsos elétricos, que constituem a mensagem entre os neurônios.

Os sinais elétricos trafegam pelos axônios e, se excederem um limiar de disparo (*threshold*), são transmitidos adiante; caso contrário, são bloqueados. A transmissão entre neurônios ocorre através de substâncias químicas, como a serotonina. A cada conexão, ou sinapse, é associado um peso, que multiplica o sinal transmitido e representa a memória do neurônio.

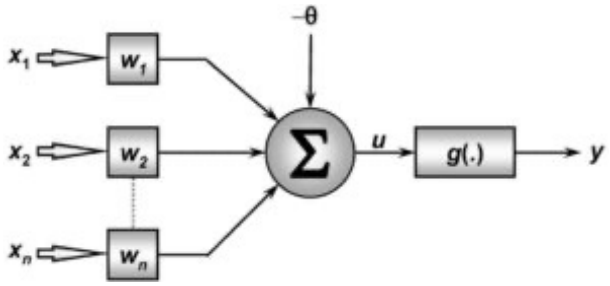
Cada região do cérebro desempenha funções específicas, como processamento auditivo, visual, e pensamento, utilizando redes interligadas que operam em paralelo. A arquitetura neural varia conforme a função, com diferenças no número de neurônios, sinapses por neurônio, valores de *threshold* e pesos sinápticos. Esses pesos são ajustados ao longo da vida, num processo conhecido como aprendizado ou memorização.

Inspirado pelo neurônio biológico, foi desenvolvido um modelo matemático de neurônio que se tornou a base da IA. Esse neurônio artificial recebe um ou mais sinais de entrada e gera um único sinal de saída, que pode ser transmitido para outros neurônios subsequentes, formando uma Rede Neural Artificial. As sinapses e axônios são representados matematicamente, com pesos sinápticos que determinam a intensidade da transmissão. O neurônio então soma todos os sinais de entrada, gerando um resultado. Esse processo é conhecido como função de combinação. A seguir, a função de ativação decide se o sinal será propagado ao longo da rede, conforme valores máximos e mínimos pré-estabelecidos.

Os componentes matemáticos envolvidos nesse processo incluem:

- **Sinais de entrada { X1, X2, Xn }**: Valores externos que alimentam o modelo.
- **Pesos sinápticos { W1, W2, Wn }**: Fatores que ponderam os sinais de entrada. Esses valores são ajustados durante o treinamento da rede.

Figura 4 – Representação Simplificada do Neurônio Matemático



Fonte: (Academy, 2018)

- **Combinador linear { Σ }**: Soma ponderada dos sinais de entrada, resultando em um potencial de ativação.
- **Limiar de ativação { Θ }**: Define o patamar necessário para gerar um sinal de ativação.
- **Potencial de ativação { v }**: Resultado da diferença entre o combinador linear e o limiar de ativação. Se $v \geq 0$, o neurônio é ativado; caso contrário, é inibido.
- **Função de ativação { g }**: Limita a saída do neurônio a um intervalo conhecido.
- **Sinal de saída { y }**: Valor final, que pode ser usado como entrada para outros neurônios subsequentes.

2.1.4 Rede Perceptron Simples e Rede Perceptron Multicamadas (MLP)

O modelo Perceptron foi desenvolvido por Frank Rosenblatt entre as décadas de 1950 e 1960 (Rosenblatt, 1957), inspirado nos trabalhos pioneiros de Warren McCulloch e Walter Pitts (Mcculloch; Pitts, 1943). O Perceptron é um modelo matemático que recebe várias entradas e gera uma única saída binária, sendo utilizado como um classificador linear em problemas de aprendizado supervisionado. Rosenblatt construiu um Perceptron de camada única, o que limitou o modelo à classificação linear e impossibilitou a modelagem hierárquica de características. Isso impediu que o Perceptron conseguisse realizar classificação não linear, como a função XOR, conforme demonstrado por Minsky e Papert (Minsky; Papert, 1969).

A Rede Perceptron Multicamadas (MLP) expande o Perceptron simples ao incluir uma ou mais camadas ocultas entre a camada de entrada e a camada de saída (Russell *et al.*, 1995). Essas camadas intermediárias permitem ao MLP modelar relações não lineares complexas. Com uma única camada oculta, as MLPs são capazes de aproximar qualquer função contínua (Haykin, 2009). MLPs são amplamente aplicados em problemas de aprendizado supervisionado, onde são treinados em conjuntos de dados rotulados para aprender a modelar a relação entre entradas e saídas.

2.1.5 Aprendizado Supervisionado e Não Supervisionado

O aprendizado supervisionado é uma abordagem de treinamento em que uma rede neural é ensinada usando um conjunto de dados rotulados. Cada exemplo de treinamento consiste em uma entrada e uma saída conhecida. O objetivo é que o modelo aprenda a mapear corretamente as entradas para as saídas, minimizando o erro entre suas previsões e os rótulos reais (Mohri; Rostamizadeh; Talwalkar, 2018).

Características do aprendizado supervisionado:

- **Objetivo**: Predizer a saída correta para novas entradas, com base no conhecimento adquirido durante o treinamento.
- **Dados Rotulados**: Utiliza um conjunto de dados de treinamento com saídas conhecidas.
- **Complexidade dos Dados**: Geralmente envolve dados menos complexos, uma vez que os rótulos guiam o aprendizado.
- **Aplicabilidade**: Classificação de imagens, reconhecimento de fala, diagnósticos médicos, previsão de preços, entre outros.

Exemplos de algoritmos de aprendizado supervisionado:

- Redes Neurais Convolucionais (CNNs).
- Redes Neurais Recorrentes (RNNs).

Por outro lado, o aprendizado não supervisionado é uma técnica em que o modelo é treinado com dados não rotulados. O objetivo é identificar padrões ou estruturas ocultas dentro dos dados. O modelo aprende diretamente da estrutura dos dados, sem informações prévias sobre as saídas desejadas (Mohri; Rostamizadeh; Talwalkar, 2018).

Características do aprendizado não supervisionado:

- **Objetivo**: Descobrir padrões, agrupamentos ou representações latentes nos dados.
- **Dados Não Rotulados**: Utiliza um conjunto de dados de treinamento sem saídas conhecidas.
- **Complexidade dos Dados**: Frequentemente aplicado a dados mais complexos e estruturados intrinsecamente, sem orientação explícita.
- **Aplicabilidade**: Agrupamento de clientes, compressão de dados, detecção de anomalias, análise de redes sociais, entre outros.

Exemplos de algoritmos de aprendizado não supervisionado:

- K-means.
- Redes Neurais Generativas Adversárias (GANs).
- Autoencoders.

Os modelos baseados em RNAs têm atraído atenção por resolverem problemas complexos de IA. A partir do conceito de neurônio matemático, diversas arquiteturas avançadas de Aprendizagem Profunda, como as Redes Neurais Convolucionais, exploradas a seguir.

2.2 Arquiteturas Avançadas de Aprendizagem Profunda

2.2.1 Visão Geral

Existem diversas arquiteturas de redes neurais, cada uma projetada para atender a finalidades específicas e resolver problemas distintos. Redes Neurais Convolucionais (CNNs), por exemplo, são amplamente empregadas em tarefas de Visão Computacional, enquanto Redes Neurais Recorrentes (RNNs) são mais adequadas para Processamento de Linguagem Natural. A Figura 5 apresenta uma visão geral das principais arquiteturas de redes neurais.

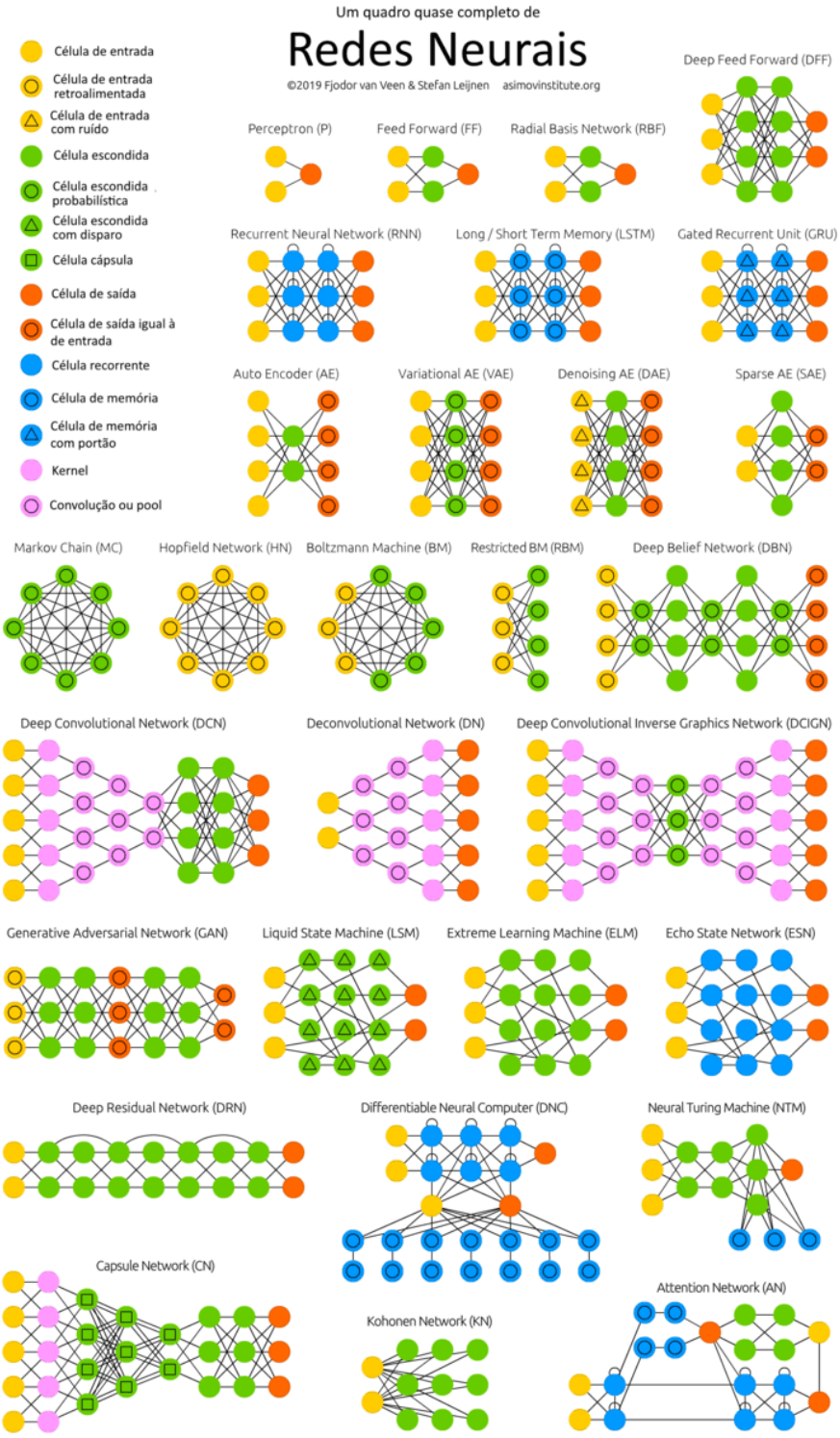
Os modelos de Aprendizagem Profunda são caracterizados pelo uso de redes neurais artificiais com múltiplas camadas ocultas ou intermediárias, como discutido por Bengio (Bengio, 2009). Nas subseções a seguir, exploraremos algumas das arquiteturas mais relevantes para os objetivos desta dissertação.

2.2.2 Redes Neurais Convolucionais (CNNs)

Uma Rede Neural Convolucional é uma classe de Rede Neural Artificial amplamente utilizada em tarefas de processamento de imagens. No livro de Chollet (Chollet, 2017), são apresentadas as quatro principais camadas de uma CNN: Convolução, *Pooling*, Camada Totalmente Conectada e Unidades Lineares Retificadas (do inglês *Rectified Linear Units* - ReLU), como ilustrado na Figura 6.

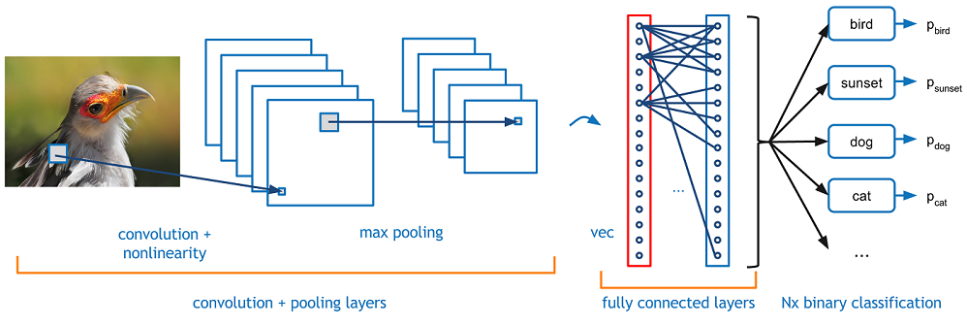
As CNNs têm demonstrado grande eficácia em tarefas de processamento de imagens, como evidenciado no trabalho de Krizhevsky, Sutskever e Hinton (Krizhevsky; Sutskever; Hinton, 2012), que apresentou a AlexNet, uma arquitetura que venceu a competição ImageNet de 2012. A importância das CNNs também foi reforçada pelo trabalho de Kaiming He, Xiangyu Zhang, Shaoqing Ren e Jian Sun (He *et al.*, 2016), que introduziram as Redes Residuais (ResNet), uma arquitetura que resolve problemas de degradação em redes muito profundas. Outro exemplo significativo é o trabalho de Karen Simonyan e

Figura 5 – Representação das Principais Redes Neurais Existentes



Fonte: (Leijnen; Veen, 2020)

Figura 6 – Arquitetura de uma rede CNN



Fonte: (Academy, 2018)

Andrew Zisserman (Simonyan; Zisserman, 2015), que apresentou a arquitetura VGG, caracterizada pelo uso de convoluções pequenas empilhadas para aumentar a profundidade e a performance das CNNs em grandes conjuntos de dados de imagens.

As CNNs possuem várias características e vantagens que as tornam extremamente eficazes em tarefas de processamento de imagens, destacando-se:

1. **Extração Automática de Características**: As CNNs utilizam camadas convolucionais para extrair automaticamente características relevantes das imagens. Esses filtros convolucionais detectam bordas, texturas, padrões e outras características importantes sem necessidade de intervenção manual. Além disso, as camadas mais profundas combinam características simples detectadas nas camadas anteriores para reconhecer formas e objetos mais complexos.

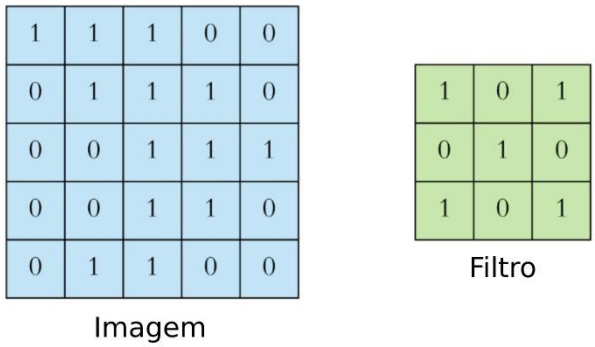
2. **Redução da Dimensionalidade**: Camadas de *Pooling* são utilizadas para reduzir a dimensionalidade dos mapas de características, o que ajuda a diminuir a complexidade computacional e a evitar o *overfitting*, mantendo as características mais importantes.

3. **Invariância a Translações e Deformações**: Devido ao uso de filtros locais aplicados em imagens, as CNNs são naturalmente invariantes a translações, rotações e pequenas deformações nas imagens, melhorando a robustez do modelo.

4. **Compartilhamento de Pesos**: Os mesmos filtros são aplicados em diferentes partes da imagem, permitindo a detecção de características independentemente da posição na imagem. Isso reduz significativamente o número de parâmetros a serem aprendidos, tornando o treinamento mais eficiente.

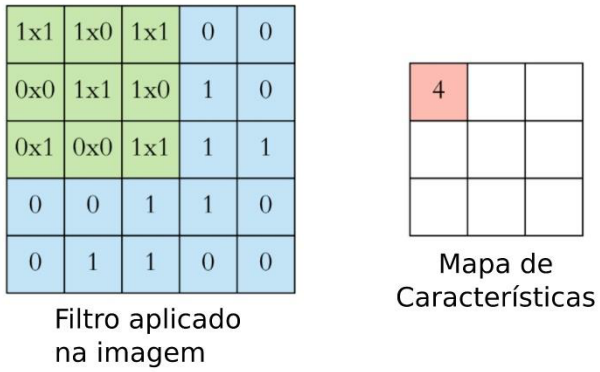
5. **Eficiência Computacional**: As operações convolucionais são eficientes e podem ser aceleradas utilizando hardware especializado, como GPUs, permitindo treinar redes profundas em grandes conjuntos de dados de imagens de maneira relativamente rápida.

Figura 7 – Exemplo de Imagem com Filtro Aplicado



Fonte: Adaptado de (DERTAT, 2017)

Figura 8 – Convolução de uma Imagem



Fonte: Adaptado de (DERTAT, 2017)

6. **Arquiteturas Profundas e Flexíveis**: Arquiteturas como AlexNet, VGG, ResNet e Inception oferecem modelos pré-treinados em grandes bases de dados como ImageNet, facilitando o uso de CNNs em diversas aplicações de processamento de imagens por meio de *transfer learning*, sem a necessidade de treinar redes do zero.

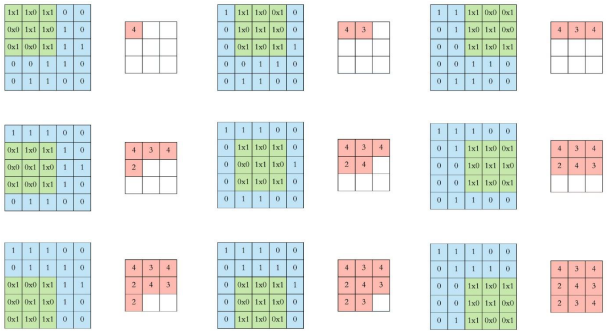
A seguir, exploraremos as quatro principais camadas de uma rede CNN.

2.2.2.1 Camadas Convolucionais

A camada convolucional é a principal responsável pela extração de características em uma CNN. A operação de convolução envolve a aplicação de um filtro (ou *kernel*) sobre a imagem. O filtro realiza uma multiplicação ponto a ponto com uma região da imagem, e os resultados dessas multiplicações são somados para produzir um único valor no Mapa de Características (ou *Feature Map*). A Figura 7 ilustra um exemplo de uma imagem 5x5 *pixels* com um filtro 3x3 aplicado.

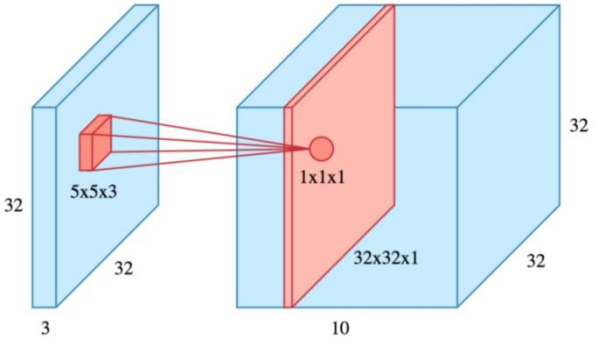
A operação de convolução é realizada deslizando o filtro por toda a imagem, resultando em um Mapa de Características. A Figura 8 demonstra o processo de convolução.

Figura 9 – Mapa de Características Utilizados na Convolução



Fonte: Adaptado de (DERTAT, 2017)

Figura 10 – Convolução de um Filtro Sendo Aplicado em uma Imagem RGB



Fonte: Adaptado de (DERTAT, 2017)

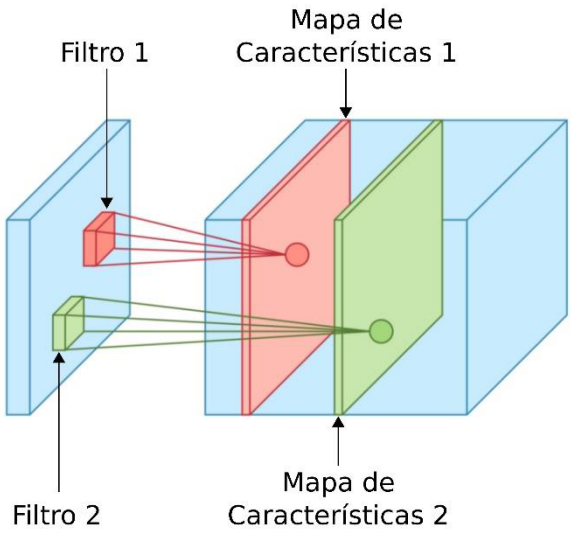
O Mapa de Características é gerado ao aplicar o filtro em cada posição da imagem. A Figura 9 mostra como esse Mapa é produzido durante o processo de convolução.

Embora os exemplos anteriores mostrem a convolução em duas dimensões (2D), as CNNs também podem operar com volumes tridimensionais (3D). A Figura 10 ilustra a aplicação de um filtro 5x5x3 em uma imagem RGB 32x32x3. A profundidade do filtro deve corresponder à profundidade da imagem na qual está sendo aplicado. Além disso, a Figura 11 demonstra como dois filtros distintos produzem dois Mapas de Características diferentes.

2.2.2.2 Camadas de *Pooling*

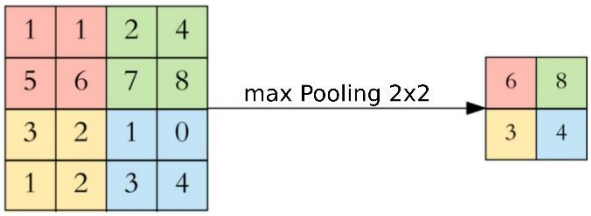
As camadas de *Pooling* têm a função principal de reduzir a dimensionalidade das ativações, o que ajuda a diminuir a complexidade computacional e a prevenir *overfitting*. O tipo mais comum de *Pooling* é o *max pooling*, que seleciona o maior valor dentro da área onde o filtro é aplicado. A Figura 12 ilustra a aplicação de um filtro *max pooling* 2x2 em uma imagem 4x4. Cada cor representa uma região diferente onde o filtro foi aplicado e seu resultado.

Figura 11 – Exemplo CNN com Dois Filtros



Fonte: Adaptado de (DERTAT, 2017)

Figura 12 – Max Pooling Aplicado em uma Imagem



Fonte: Adaptado de (DERTAT, 2017)

2.2.2.3 Camadas Totalmente Conectadas

As camadas Totalmente Conectadas (*Fully Connected*) são responsáveis por processar as ativações finais da CNN e transformá-las em um vetor. Após a última camada convolucional, os dados são achatados em um vetor e processados por uma ou mais camadas totalmente conectadas. A saída da última camada totalmente conectada é um vetor com uma dimensão igual ao número de classes no problema de classificação, que é então usado pelo classificador para gerar a predição final.

2.2.2.4 Aprendizado por Transferência

Quando não há uma base de dados suficientemente grande para treinar uma CNN do zero, técnicas de aprendizado por transferência podem ser usadas. Essas técnicas aproveitam modelos pré-treinados em grandes bases de dados para auxiliar na classificação de novas bases de imagens. De acordo com Li, Wu e Gao (Li; Wu; Gao, 2023), as duas principais técnicas de aprendizado por transferência são o Ajuste Fino (*fine tuning*) e a Extração de Características (*feature extraction*).

Na técnica de Ajuste Fino, uma CNN pré-treinada é retreinada com uma nova base

de imagens. Esta abordagem é baseada na premissa de que as características extraídas pela rede pré-treinada são úteis também para a nova tarefa, necessitando apenas de ajustes nos pesos das camadas finais. A vantagem dessa técnica é que as primeiras camadas, que capturam características mais gerais, já estão treinadas, permitindo que o treinamento se concentre nas últimas camadas que são mais específicas para a nova base de dados. Isso reduz a necessidade de um grande número de amostras e o tempo de treinamento.

Na técnica de Extração de Características, a CNN pré-treinada é utilizada apenas como um extrator de características. As ativações de qualquer camada podem ser transformadas em vetores de características para serem usados por um classificador, geralmente utilizando as ativações das camadas totalmente conectadas, que já são vetores. Esta técnica é aplicada quando se tem uma base de dados muito pequena, tornando inviável o uso de Ajuste Fino, ou quando não se dispõe de recursos computacionais robustos para treinar uma CNN.

2.2.3 Funções de Ativação

A função de ativação é um componente matemático crucial nas Redes Neurais Artificiais (RNAs) que possibilita a resolução de problemas complexos. De acordo com Glorot, Xavier e Yoshua (Glorot; Bengio, 2010), várias funções de ativação, incluindo Sigmoide, Tanh e ReLU, são amplamente utilizadas e discutidas em termos de suas vantagens e desvantagens em redes profundas.

2.2.3.1 Sigmoide

A função Sigmoide é amplamente utilizada como função de ativação e é definida pela fórmula:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

A principal característica da função Sigmoide é sua não linearidade, o que permite que redes com múltiplos neurônios ativados pela função Sigmoide produzam saídas não lineares. A função varia entre 0 e 1 e possui um formato de "S". Contudo, a função Sigmoide apresenta alguns problemas, como a saturação dos gradientes, onde os gradientes se aproximam de zero, dificultando o aprendizado da rede. Outro problema que a função Sigmoide possui é que os valores variam apenas entre 0 a 1.

2.2.3.2 Tanh

A função Tanh é similar à função Sigmoide, mas com uma variação que escala a saída para o intervalo de -1 a 1. A função Tanh é definida por:

$$\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.2}$$

A função Tanh resolve o problema da saturação dos valores de saída ao permitir que os valores variem entre -1 e 1. Isso ajuda a centralizar os dados em torno de zero e pode melhorar o desempenho da rede ao evitar que todas as saídas tenham o mesmo sinal. Assim como a Sigmoide, a função Tanh é contínua e diferenciável em todos os pontos.

2.2.3.3 ReLU

A função ReLU, ou Unidade Linear Retificada, é definida por:

$$\text{ReLU}(x) = \max(0, x) \tag{2.3}$$

ReLU é amplamente utilizada em projetos de redes neurais devido à sua simplicidade e eficiência. A principal vantagem de utilizar a função ReLU sobre outras funções de ativação é que ela não ativa todos os neurônios no mesmo instante. Isso significa que apenas alguns neurônios são ativados, tornando a rede mais eficiente. Contudo, ReLU também pode apresentar problemas com os gradientes que se deslocam em direção a 0.

2.2.4 Métricas

Nesta subseção, são apresentadas as métricas de classificação consideradas para avaliar o desempenho das redes neurais escolhidas. As fórmulas para o cálculo dessas métricas foram obtidas da obra de Hackeling (Hackeling, 2017). Para facilitar o entendimento, as siglas utilizadas nas fórmulas são definidas a seguir:

- **TP (do inglês *True Positives*)**: Exemplos positivos corretamente classificados como positivos pelo modelo.
- **FN (do inglês *False Negatives*)**: Exemplos positivos incorretamente classificados como negativos pelo modelo.
- **FP (do inglês *False Positives*)**: Exemplos negativos incorretamente classificados como positivos pelo modelo.
- **TN (do inglês *True Negatives*)**: Exemplos negativos corretamente classificados como negativos pelo modelo.

2.2.4.1 Acurácia

A acurácia mede a proporção de previsões corretas feitas pelo modelo em relação ao número total de previsões. A fórmula para o cálculo da acurácia é:

$$\text{Acurácia} = \frac{\text{Número de predições corretas}}{\text{Número total de predições}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.4}$$

2.2.4.2 Precisão

A precisão (ou *precision*) é a razão entre o número de verdadeiros positivos e o total de previsões positivas feitas pelo modelo. A fórmula para o cálculo da precisão é:

$$\text{Precisão} = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos + Falsos Positivos}} = \frac{TP}{TP + FP} \tag{2.5}$$

2.2.4.3 Cobertura

A cobertura (ou *recall*) é a razão entre o número de verdadeiros positivos e o total de exemplos que realmente pertencem à classe positiva. A fórmula para o cálculo da cobertura é:

$$\text{Cobertura} = \frac{\text{Verdadeiros positivos}}{\text{Verdadeiros positivos + Falsos negativos}} = \frac{TP}{TP + FN} \tag{2.6}$$

2.2.4.4 *Log Loss*

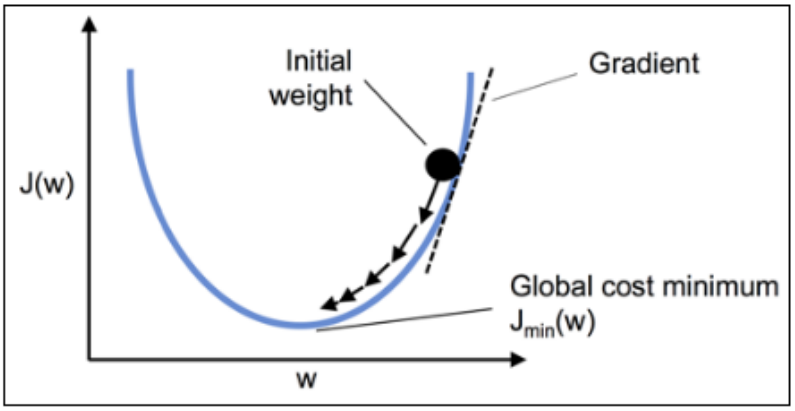
Log Loss é uma função de perda utilizada para avaliar a performance de modelos de classificação, especialmente em redes neurais artificiais (RNAs). A fórmula da *log loss* é:

$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)] \tag{2.7}$$

- N: Número total de exemplos.
- y_i: Valor verdadeiro da classe para o i-ésimo exemplo (1 se for a classe positiva, 0 se for a classe negativa).
- p_i: Probabilidade prevista de que o i-ésimo exemplo pertença à classe positiva.

Entre as métricas selecionadas, a *log loss* é de especial relevância, pois as redes neurais buscam minimizar essa métrica durante o processo de treinamento, refletindo diretamente na precisão das previsões.

Figura 13 – Gradiente Descendente



Fonte: (Raschka; Mirjalili, 2017)

2.2.4.5 *Overhead*

O cálculo do *overhead* envolve uma combinação de medições práticas, análise de código e entendimento das operações adicionais específicas que estão sendo realizadas. Esta métrica é crucial para otimizar a eficiência da RNA e garantir um desempenho adequado para as aplicações desejadas. A fórmula do *overhead* é:

$$\text{Overhead} = \frac{\text{Tempo Total de Execução}}{\text{Tempo de Execução Útil}} \tag{2.8}$$

2.2.5 Otimizadores

Um dos objetivos dos algoritmos de aprendizado de máquina supervisionados é otimizar o processo de redução da função de custo, também chamada de função J (Raschka; Mirjalili, 2017). Reduzir a função J é fundamental para otimizar a rede, permitindo que ela identifique os pesos que melhor representam a relação entre os dados. Esses pesos formam o modelo preditivo, que possibilita à rede fazer predições ao utilizar novos conjuntos de dados. Para descobrir esses pesos, a rede é treinada para fazer predições o mais próximas possíveis dos valores reais. A função de custo *log loss* é utilizada para medir o quão erradas são as previsões:

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{2.9}$$

O algoritmo de otimização para encontrar os pesos é o Gradiente Descendente (GD). A Figura 13 demonstra que, para cada iteração, é dado um passo na direção oposta ao gradiente, onde o tamanho do passo é determinado pela taxa de aprendizado (Raschka; Mirjalili, 2017).

Com a descida do gradiente, são realizados pequenos passos em direção ao mínimo global. Nesse processo, a rede ajusta os pesos em etapas que reduzem o erro. Como o caminho mais rápido está na direção mais íngreme, as etapas tomadas devem estar na direção que minimiza o erro. O GD atualiza os pesos, dando um passo na direção oposta ao gradiente $\Delta J(w)$ da função de custo J:

$$w := w + \Delta w \tag{2.10}$$

Onde a mudança de peso $\Delta J(w)$ é definida como o gradiente negativo multiplicado pela taxa de aprendizado $-\eta$:

$$\Delta w = -\eta \Delta J(w) \tag{2.11}$$

Para calcular o gradiente da função de custo J, é necessário calcular a derivada da função de custo em relação a cada peso wj. Para isso, as redes utilizam um algoritmo chamado *Backpropagation*, que otimiza o processo de cálculo das derivadas, tornando a rede mais eficiente. Embora o GD apresente bons resultados, ele possui custos elevados para atualizar os pesos quando o conjunto de dados é muito grande, pois é preciso reavaliar todo o conjunto de dados de treinamento para cada passo em direção ao mínimo global (Raschka; Mirjalili, 2017).

Uma alternativa ao GD é o Gradiente Descendente Estocástico (do inglês *Stochastic Gradient Descent* - SGD), que, ao invés de atualizar os pesos com base na soma dos erros acumulados sobre todos os dados, aplica o GD a amostras aleatórias de dados de treinamento. A vantagem é que a convergência da rede é alcançada mais rapidamente através de pequenos lotes, devido às atualizações de peso serem mais frequentes (Raschka; Mirjalili, 2017). Os parâmetros do SGD que ajudam no treinamento da rede são:

- **Taxa de aprendizado**: Indica a velocidade com que o otimizador ajusta os pesos da rede neural.
- **Momentum**: Refere-se à quantidade de inércia que o gradiente acumula, ajudando a suavizar o processo de atualização dos pesos.
- **Decay**: Utilizado para reduzir gradualmente a taxa de aprendizado conforme o treinamento avança.

Além dos otimizadores já mencionados, o AdamW combina os benefícios do método de descida de gradiente adaptativo, ajustando a taxa de aprendizado para cada parâmetro com base nas estimativas dos momentos de primeira e segunda ordem, e o decaimento de peso (*weight decay*), que é implementado de forma mais direta e eficiente do que no

Adam tradicional (Loshchilov; Hutter, 2018). O AdamW utiliza um decaimento de peso explícito, que age como regularizador, penalizando grandes valores de pesos e prevenindo o sobreajuste. Essa abordagem tem se mostrado vantajosa em termos de generalização, pois melhora o desempenho em conjuntos de dados complexos e em redes neurais profundas (Loshchilov; Hutter, 2018).

Nesta dissertação, foram considerados os otimizadores SGD e AdamW, com o objetivo de determinar qual dos dois é mais eficaz na classificação de imagens faciais.

2.3 Processamento de Imagem com CNNs

Neste capítulo, exploraremos o funcionamento do processamento de imagens utilizando redes convolucionais (CNNs), desde as etapas de pré-processamento, com técnicas de padronização de dados e *data augmentation*, até a extração de características, passando por arquiteturas avançadas.

2.3.1 Pré-processamento de Imagens

2.3.1.1 Normalização e Padronização de Dados

A normalização e a padronização são técnicas essenciais no pré-processamento de imagens para CNNs, permitindo que os dados de entrada estejam em uma escala adequada para o treinamento eficiente dos modelos.

Segundo LeCun et al. (LeCun *et al.*, 1998), a técnica de normalização refere-se ao ajuste dos valores dos *pixels* para um intervalo específico, geralmente entre 0 e 1. Essa técnica reduz a disparidade entre os valores dos *pixels*, facilitando a convergência dos algoritmos de aprendizado de máquina. A normalização pode ser realizada dividindo os valores dos *pixels* pelo valor máximo possível.

Por outro lado, Ioffe e Szegedy (Ioffe; Szegedy, 2015) destacam que a técnica de padronização envolve transformar os dados para que tenham média zero e variância unitária. Isso é particularmente útil para algoritmos que assumem uma distribuição normal dos dados de entrada. A padronização é realizada subtraindo-se a média e dividindo pelo desvio padrão dos valores dos *pixels*.

2.3.1.2 Técnicas de *Data Augmentation*

As técnicas de *data augmentation* têm como objetivo aumentar o conjunto de dados de treinamento, criando variações artificiais das imagens de entrada. Isso contribui para melhorar a generalização dos modelos e prevenir o *overfitting*. Abaixo, são descritas algumas formas de realizar essa ampliação dos dados.

Simard et al. (Simard; Steinkraus; Platt, 2003) discutem a técnica de rotação e translação, que consiste em aplicar pequenas rotações e translações às imagens, ajudando o

modelo a aprender a reconhecer objetos, como faces, independentemente de sua orientação ou posição.

Krizhevsky et al. (Krizhevsky; Sutskever; Hinton, 2012) introduzem a técnica de espelhamento e inversão, onde imagens espelhadas horizontalmente são incluídas no conjunto de dados, visando ensinar o modelo a reconhecer faces em diferentes direções.

Howard (Howard, 2013) propõe a técnica de mudanças de iluminação, que altera a intensidade da luz e a exposição nas imagens para tornar o modelo mais robusto a diferentes condições de iluminação.

Por fim, Bishop (Bishop, 1995) apresenta a técnica de adição de ruído, que consiste em adicionar ruído às imagens para melhorar a robustez do modelo contra interferências e imperfeições nas imagens de entrada.

2.3.1.3 Técnicas de *Undersampling*

A técnica de undersampling é amplamente utilizada em aprendizado de máquina para lidar com conjunto de dados desbalanceados, onde a presença dominante de uma ou mais classes pode causar vieses nos modelos, resultando em um desempenho insatisfatório para as classes minoritárias. O método mais simples, o *undersampling* aleatório, reduz o número de instâncias da classe majoritária para equilibrar a distribuição entre as classes, mas pode levar à perda de informações cruciais, nos estudos de Gustavo e Ronaldo (Batista; Prati; Monard, 2004).

No reconhecimento facial, onde o desbalanceamento demográfico é comum, o *undersampling* pode ajudar a evitar que modelos favoreçam grupos majoritários, contribuindo para resultados mais justos. No entanto, a aplicação dessa técnica deve ser cuidadosa, pois a remoção inadequada de dados pode introduzir vieses negativos e reduzir a capacidade de generalização do modelo.

2.3.2 Extração de Características

A extração de características é o processo de identificar e quantificar aspectos relevantes das imagens, que são usados pelos modelos de aprendizado para fazer previsões. Este processo envolve a identificação de padrões, bordas, texturas e outras características visuais importantes que podem ser usadas para diferenciar entre diferentes classes de imagens, conforme explorado por Turk e Pentland (Turk; Pentland, 1991).

A detecção de bordas, texturas e padrões é uma parte crucial na extração de características, permitindo que o modelo capture detalhes essenciais da imagem:

- **Detecção de Bordas**: Métodos como *Sobel*, *Canny* e *Laplacian* são utilizados para identificar contornos e formas nas imagens, ajudando a isolar objetos de interesse do fundo.

- **Detecção de Texturas**: Filtros de *Gabor* e transformadas *wavelet* são comumente usados para capturar texturas em imagens, identificando padrões repetitivos ou estruturados que distinguem diferentes superfícies e materiais.

- **Detecção de Padrões**: A transformada de *Fourier* e a Análise de Componentes Principais (PCA) são técnicas usadas para identificar padrões repetitivos nas imagens, auxiliando na simplificação de dados complexos e na extração de características globais.

2.3.3 Arquiteturas Avançadas

A seguir, são exploradas algumas arquiteturas que marcaram importantes avanços no desenvolvimento de CNNs, cada uma trazendo inovações que melhoraram significativamente o desempenho em tarefas de reconhecimento de imagens:

- **LeNet (LeCun *et al.*, 1998)**: Uma das primeiras CNNs, desenvolvida para o reconhecimento de dígitos manuscritos, estabelecendo as bases para o uso de redes convolucionais em tarefas de visão computacional.

- **AlexNet (Krizhevsky; Sutskever; Hinton, 2012)**: Introduziu o uso de ReLU e *dropout*, melhorando significativamente a precisão em grandes conjuntos de dados, como o ImageNet, e tornando-se um marco no campo de redes profundas.

- **VGGNet (Simonyan; Zisserman, 2015)**: Caracteriza-se pelo uso de muitas camadas convolucionais pequenas (3x3) para capturar características complexas, oferecendo uma estrutura mais uniforme e simplificada, que facilitou a exploração de redes mais profundas.

- **ResNet (He *et al.*, 2016)**: Introduziu conexões residuais, que permitem a criação de redes muito profundas sem sofrer com o problema do *vanishing gradient*, possibilitando a construção de redes com centenas de camadas sem perda significativa de desempenho.

Neste trabalho, foi considerada a LResNet50E-IR para reconhecimento facial, pelos seguintes fatores:

- **Arquitetura ResNet**: A LResNet50E-IR é baseada na arquitetura ResNet, que introduz conexões residuais para mitigar o problema do desaparecimento do gradiente em redes muito profundas. O número '50' indica que essa rede possui 50 camadas, proporcionando uma profundidade considerável que permite a captura de características complexas em imagens faciais.

- **Desempenho em Reconhecimento Facial**: Redes baseadas em ResNet, como a LResNet50E-IR, são amplamente reconhecidas por seu desempenho superior em tarefas de reconhecimento facial. Elas são capazes de aprender representações faciais robustas e discriminativas, resultando em alta precisão em diversas condições, como variações de iluminação, ângulos e expressões, conforme explorado no estudo de Deng et al. (Deng *et al.*, 2019).

- **Eficiência Computacional**: Apesar de sua profundidade, a LResNet50E-IR é relativamente eficiente em termos computacionais. Ela oferece um bom equilíbrio entre precisão e uso de recursos, crucial para aplicações em tempo real, como autenticação biométrica, explorado no estudo de Zhang et al. (Zhang *et al.*, 2018).

- **Adaptação e Customização**: A LResNet50E-IR pode ser facilmente adaptada ou ajustada para diferentes contextos de aplicação, seja em termos de adaptação ao hardware disponível ou customização para um conjunto de dados específicos, garantindo flexibilidade para diferentes projetos, conforme discutido por Cao et al. (Cao *et al.*, 2018).

No contexto de redes neurais convolucionais das arquiteturas avançadas é adicionada a função de perda CrossEntropyLoss, amplamente utilizada para medir a divergência entre a distribuição prevista e a distribuição real, facilitando o treinamento de modelos de classificação (Sterr, 2020).

2.4 Reconhecimento de Biometria Facial

2.4.1 Introdução à Biometria Facial

A biometria facial refere-se ao uso de características físicas e comportamentais do rosto para o reconhecimento automático de indivíduos. Este método de identificação biométrica é amplamente utilizado devido à sua conveniência e precisão, conforme explorado no artigo de Jain et al. (Jain; Ross; Prabhakar, 2004).

O reconhecimento facial é aplicado em diversas áreas, destacando-se:

- **Segurança**: Utilizado em vigilância, monitoramento e controle de acesso, garantindo a identificação e o rastreamento de indivíduos em tempo real, como discutido no livro de Zhao et al. (Zhao *et al.*, 2003).

- **Autenticação**: Implementado em dispositivos móveis, sistemas bancários e outras plataformas para autenticar usuários, garantindo segurança e conveniência, conforme explorado por Frischholz et al. (Frischholz; Dieckmann, 2000).

2.4.2 Técnicas Clássicas vs Técnicas Baseadas em CNN

Antes do advento das CNNs, diversos métodos clássicos eram amplamente utilizados para reconhecimento facial, cada um com suas características e limitações:

- ***Eigenfaces* (Turk; Pentland, 1991)**: Utiliza a Análise de Componentes Principais (PCA) para reduzir a dimensionalidade das imagens, representando rostos em um espaço de características de menor dimensão, facilitando a distinção entre diferentes identidades faciais.
- ***Fisherfaces* (Belhumeur; Hespanha; Kriegman, 1997)**: Baseado na Análise Discriminante Linear (LDA), este método maximiza a separação entre classes enquanto minimiza a variação dentro das classes, tornando-o mais robusto em comparação com *Eigenfaces*.
- **LBPH (*Local Binary Patterns Histogram*) (Ahonen; Hadid; Pietikäinen, 2006)**: Utiliza padrões binários locais para capturar texturas e características locais do rosto, sendo especialmente eficaz em condições de variação de iluminação.

Com o advento das CNNs, o reconhecimento de biometria facial evoluiu significativamente, oferecendo vantagens substanciais:

- **Precisão**: CNNs permitem a extração de características mais complexas e discriminativas das imagens faciais, melhorando a precisão do reconhecimento (Krizhevsky; Sutskever; Hinton, 2012).
- **Robustez**: Alta assertividade mesmo com variações de iluminação, pose e expressão facial, como demonstrado por Parkhi et al. (Parkhi; Vedaldi; Zisserman, 2015).
- **Escalabilidade**: As CNNs podem ser treinadas em grandes conjuntos de dados, permitindo a construção de modelos robustos e escaláveis (LeCun; Bengio; Hinton, 2015).

2.4.3 Modelos Avançados

A seguir, são explorados os modelos avançados de reconhecimento facial, com uma descrição de seu funcionamento e suas vantagens.

2.4.3.1 FaceNet

O FaceNet, desenvolvido pelo Google (Schroff; Kalenichenko; Philbin, 2015), utiliza uma CNN para aprender uma representação embutida de imagens faciais. A abordagem de *triplet loss* agrupa imagens faciais em tripletos (âncora, positivo e negativo) para aprender

uma métrica de similaridade. Isso permite que o FaceNet projete imagens faciais em um espaço de características de alta dimensão, onde distâncias euclidianas entre vetores correspondem a similaridades faciais.

As principais vantagens deste modelo incluem sua alta precisão em *benchmarks* e a eficiência na comparação rápida de características faciais, tornando-o ideal para sistemas de reconhecimento em larga escala (Schroff; Kalenichenko; Philbin, 2015).

2.4.3.2 DeepFace

O DeepFace, desenvolvido pelo Facebook (Taigman *et al.*, 2014), utiliza uma rede neural convolucional profunda para aprender representações faciais. Com várias camadas convolucionais e camadas totalmente conectadas, o modelo é treinado em um grande conjunto de dados de rostos e utiliza uma técnica de alinhamento facial para melhorar a precisão do reconhecimento.

Suas principais vantagens são a alta precisão em condições de variação de pose e iluminação, e a técnica de alinhamento facial, que aumenta a robustez do modelo em diferentes condições de entrada (Taigman *et al.*, 2014).

2.4.3.3 VGG-Face

O VGG-Face, desenvolvido pelo Visual Geometry Group (Parkhi; Vedaldi; Zisserman, 2015) da Universidade de Oxford, é baseado na arquitetura VGGNet. Este modelo é treinado com um grande conjunto de dados de imagens faciais e utiliza uma rede convolucional profunda com camadas uniformes de convolução e *pooling*. A arquitetura do VGG-Face é conhecida por sua simplicidade e eficácia em extrair características faciais detalhadas.

As principais vantagens deste modelo incluem sua robustez, capturando características faciais detalhadas e robustas, e sua flexibilidade, permitindo fácil adaptação para diferentes tarefas de reconhecimento facial (Parkhi; Vedaldi; Zisserman, 2015).

2.4.3.4 ArcFace

O ArcFace foi projetado para superar as limitações dos modelos anteriores, especialmente em termos de discriminação e robustez em reconhecimento facial (Deng *et al.*, 2019).

Os desafios dos modelos anteriores incluem:

- **FaceNet**: A perda de *triplet* pode não capturar suficientemente a separação entre classes faciais, especialmente em cenários com grande variação.

- **DeepFace**: Embora a técnica de alinhamento melhore a precisão, a rede pode enfrentar dificuldades com variações extremas de pose e iluminação.
- **VGG-Face**: Embora seja detalhado, o modelo pode não oferecer a separação angular necessária para distinguir identidades faciais de maneira robusta.

O ArcFaceLoss é a técnica avançada projetada para melhorar o desempenho do reconhecimento facial, introduzindo um termo de margem angular na função de perda. Isso resulta em representações faciais mais discriminativas e, consequentemente, em uma maior acurácia no reconhecimento.

Nesta dissertação, foi utilizado a Arquitetura LResNet50E-IR com ArcFace e a técnica ArcFaceLoss, que se destaca pelas seguintes características:

- **Perda Angular**: Introduz a técnica de perda angular, que melhora a separação entre identidades faciais ao penalizar a proximidade angular de vetores de características. Isso resolve o problema de discriminação que os modelos anteriores não abordavam de forma tão eficaz.
- **Desempenho Melhorado**: A abordagem de margem angular aditiva do ArcFace permite que ele alcance melhores resultados em *benchmarks* de reconhecimento facial, superando os modelos anteriores em termos de precisão e robustez (Deng *et al.*, 2019).

2.5 Implementação Prática

Neste capítulo final, são exploradas as técnicas de implementação de reconhecimento facial, detalhando um pipeline estruturado que abrange as etapas de detecção, alinhamento, extração de características e reconhecimento. As ferramentas utilizadas incluem TensorFlow, Keras, PyTorch e OpenCV.

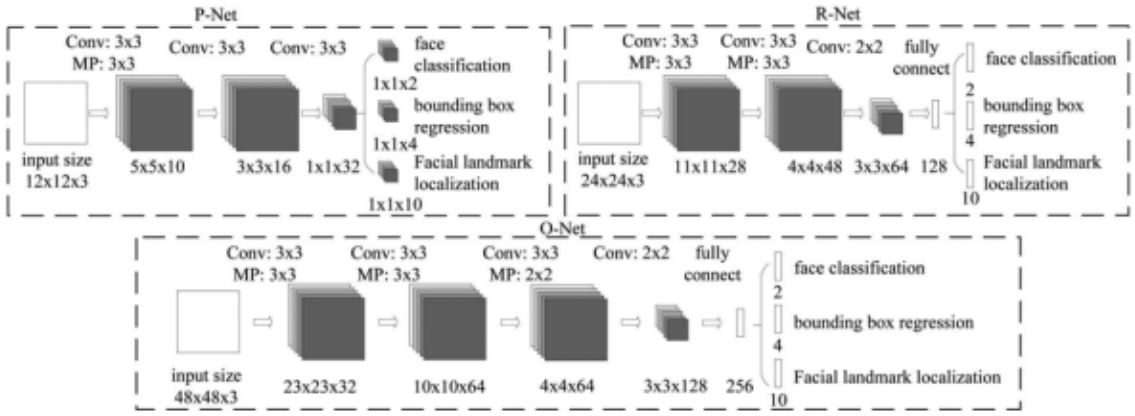
2.5.1 Detecção e Alinhamento Facial

A detecção facial é a primeira etapa do pipeline de reconhecimento facial, cujo objetivo é localizar e extrair as regiões do rosto em uma imagem.

O método adotado nesta dissertação foi o MTCNN (*Multi-task Cascaded Convolutional Networks*), conforme proposto nos trabalhos de Zhang et al. (Zhang *et al.*, 2016) e Sun et al. (Sun; Wang; Tang, 2016). O MTCNN é um conjunto de modelos de *Deep Learning* baseados em redes neurais convolucionais (CNN) projetados para realizar detecção de faces em imagens. Ele opera em três estágios hierárquicos: P-Net, R-Net e O-Net, cuja arquitetura está ilustrada na Figura 14.

Resumidamente, essa arquitetura possui três saídas principais:

Figura 14 – Arquitetura de P-Net, R-Net, e O-Net



Fonte: (Zhang *et al.*, 2016)

- **Bounding Boxes**: Caixas delimitadoras que indicam onde as faces estão localizadas na imagem. As *bounding boxes* são normalizadas para garantir a consistência e facilitar a comparação entre diferentes escalas de imagens.
- **Níveis de Confiança (*Scores*)**: Valores que indicam o nível de confiança do modelo de que a caixa gerada corresponde a uma face.
- **Pontos-Chave Faciais**: Coordenadas dos pontos-chave faciais, como olhos, nariz e boca, que são cruciais para o alinhamento facial.

2.5.2 Ferramentas e Frameworks

Abaixo, são descritas as ferramentas e *frameworks* selecionados para o desenvolvimento desta dissertação, sendo essas as principais bibliotecas de *Deep Learning* usadas na implementação de modelos de reconhecimento facial.

- **PyTorch**: Desenvolvida pelo Facebook, é uma biblioteca de *deep learning* amplamente reconhecida por sua flexibilidade e eficiência tanto em pesquisa quanto em produção (Paszke *et al.*, 2019).
- **OpenCV (*Open Source Computer Vision Library*)**: Biblioteca amplamente utilizada para tarefas de visão computacional, incluindo o pré-processamento de imagens para reconhecimento facial (Bradski, 2000). Fornece diversas funções para detecção e alinhamento facial, além de outras operações de pré-processamento, como redimensionamento, rotação e filtragem de imagens.
- **TensorFlow**: Desenvolvida pelo Google, é uma biblioteca de código aberto para aprendizado de máquina, amplamente usada para construir e treinar redes neurais profundas (Abadi *et al.*, 2016).

- **Keras**: API de alto nível para redes neurais que funciona sobre o TensorFlow, simplificando a construção e o treinamento de modelos (Chollet *et al.*, 2015).

3 METODOLOGIA

Para alcançar os objetivos desta dissertação, será adotado o processo de Extração de Conhecimento, também conhecido como KDD (*Knowledge Discovery in Databases*). Este processo envolve uma série de etapas essenciais para extrair e validar informações valiosas a partir de bases de dados (Fayyad; Piatetsky-Shapiro; Smyth, 1996).

3.1 Seleção e Análise do Conjunto de Dados

O objetivo desta dissertação é desenvolver modelos de redes neurais para o reconhecimento de faces com ênfase na mitigação de viés. Para isso, é crucial garantir a equidade do conjunto de dados utilizado. Inicialmente, serão selecionadas Redes Neurais Convolucionais (CNNs) para a classificação de imagens e comparação com alguns conjuntos de dados. Para tanto, é fundamental buscar conjuntos que incluam imagens representativas de diversas características faciais humanas. O conjunto de dados selecionado para este estudo é:

- **FairFace**: Contendo mais de 100.000 imagens de 7.000 indivíduos, o FairFace inclui anotações detalhadas sobre atributos demográficos como raça, gênero e idade. Este conjunto visa fornecer uma base de dados mais equilibrada e representativa para treinar e avaliar algoritmos de reconhecimento facial, contribuindo para a mitigação de viés e aprimoramento da precisão em diferentes grupos demográficos (Karkkainen; Joo, 2021).

A análise do conjunto de dados será tanto quantitativa quanto qualitativa. A avaliação quantitativa envolve a análise estatística das métricas de desempenho dos modelos, como acurácia, precisão e *recall*. A avaliação qualitativa, por outro lado, examinará a capacidade dos modelos em lidar com a diversidade e complexidade dos dados, observando como o viés é mitigado e se há uma melhora na generalização do reconhecimento facial em diferentes grupos demográficos.

Ainda sobre a avaliação qualitativa, serão consideradas as 7 raças presentes no conjunto de dados FairFace:

- **Black**: Refere-se a indivíduos com ascendência africana ou afrodescendente, incluindo pessoas de diversas regiões da África, América Latina e outros lugares, com uma ampla variedade de tons de pele e características faciais.
- **East Asian**: Inclui pessoas de ascendência predominantemente asiática oriental, com origem em países como China, Japão, Coreia e Mongólia, entre outros. Caracte-

rísticas faciais comuns podem incluir formato de olhos com dobras epicânticas e pele geralmente clara a moderada.

- **Indian**: Refere-se a indivíduos do subcontinente indiano, incluindo Índia, Paquistão, Bangladesh, Sri Lanka e áreas adjacentes. Pessoas deste grupo podem apresentar tons de pele variados e características faciais associadas a essas regiões.

- **Latino Hispanic**: Engloba indivíduos com ascendência latino-americana ou hispânica, incluindo regiões da América Latina, Caribe e áreas de língua espanhola. As características variam bastante, refletindo a diversidade genética da região.

- **Middle Eastern**: Refere-se a pessoas de origem no Oriente Médio, incluindo regiões como Arábia Saudita, Irã, Iraque, Síria, Egito e países adjacentes. Indivíduos desse grupo possuem características variadas, com tons de pele de claro a moderado.

- **Southeast Asian**: Inclui indivíduos de origem do sudeste asiático, de países como Tailândia, Vietnã, Indonésia, Filipinas, Malásia e outros. Esse grupo apresenta uma ampla diversidade de tons de pele e características faciais.

- **White**: Refere-se a pessoas com ascendência europeia, incluindo regiões da Europa, América do Norte e outras áreas. Características faciais podem variar, com tons de pele geralmente claros a moderados.

3.2 Pré-Processamento dos Dados

O pré-processamento dos dados é essencial para garantir a equidade e a eficácia dos modelos. Nesta etapa, as imagens serão normalizadas utilizando o algoritmo MTCNN para detectar caixas delimitadoras dos rostos e marcos faciais, como olhos, nariz e boca. As imagens serão então cortadas, alinhadas (por meio de transformação de similaridade) e redimensionadas para 224 $\times$ 224 pixels. O conjunto de dados será dividido em partes para treino, validação e teste. Além disso, será aplicado um balanceamento no conjunto de dados, utilizando a técnica de *undersampling* para a menor classe.

A implementação do modelo utilizará a linguagem de programação *Python* (Foundation, 2023) e as bibliotecas *Keras* e *PyTorch*. Essas bibliotecas oferecem modularidade e extensibilidade, suportando CNNs tanto em CPU (*Central Processing Unit*) quanto em GPU (*Graphics Processing Unit*) (Chollet *et al.*, 2015). As imagens, originalmente em formato RGB, terão seus valores de pixels normalizados para *mean=[0.485, 0.456, 0.406]* e *std=[0.229, 0.224, 0.225]*. Para equilibrar o desempenho e a complexidade computacional, será adotada a arquitetura LResNet50E-IR, uma variante do ResNet, projetada com ArcFace.

3.3 Treinamento do Modelo

Durante o treinamento, serão definidos os pesos das redes e os parâmetros de treinamento, como o número de épocas e o tamanho dos lotes (*batch size*). O tamanho do *batch* será ajustado para 128. Serão utilizados os otimizadores SGD e AdamW. Para agendamento das épocas, a escolha do *scheduler* é crucial para o desempenho do modelo, portanto será considerada as duas técnicas destacadas abaixo.

- **OneCycleLR**: O OneCycleLR ajusta a taxa de aprendizado ao longo de um ciclo de treinamento, começando com um aumento gradual até um pico e, em seguida, diminuindo rapidamente, conforme descrito por Smith (Smith, 2017). Essa estratégia ajuda a evitar mínimos locais e melhora a convergência.
- **CosineAnnealingWarmRestarts**: O CosineAnnealingWarmRestarts é uma abordagem que utiliza um padrão cosseno para resfriar a taxa de aprendizado, reiniciando em intervalos regulares, o que também pode ser benéfico para a exploração e a convergência do modelo, como destacado por Loshchilov e Hutter (Loshchilov; Hutter, 2017).

3.4 Análise de Desempenho do Modelo Treinado

Após o treinamento, serão avaliadas as métricas de desempenho das redes, incluindo *recall*, acurácia, precisão, *overhead*, *F1-Score* e *log loss*. Essas métricas são cruciais para problemas de classificação. A rede com menor *log loss* e menor *overhead* será selecionada como a mais eficiente para o reconhecimento facial, considerando tanto o conjunto de dados equilibrado quanto a mitigação de viés.

A acurácia no reconhecimento de imagens pode variar dependendo do conjunto de dados e da arquitetura do modelo. Abaixo estão alguns *benchmarks* modernos em reconhecimento facial:

- LFW (Labeled Faces in the Wild): Um dos benchmarks mais tradicionais para reconhecimento facial é o LFW. Modelos modernos alcançam acurácias superiores a 99% neste conjunto de dados. Por exemplo, o FaceNet alcançou uma acurácia de 99.63% (Schroff; Kalenichenko; Philbin, 2015).
- MegaFace: Este é um benchmark mais desafiador que avalia a capacidade do modelo em reconhecer faces em grandes conjuntos de dados. Modelos de última geração com o ArcFace, alcançam acurácias em torno de 97% neste conjunto de dados (Deng *et al.*, 2019).
- CASIA-WebFace: Outro benchmark importante, com o VGG-Face alcançando uma acurácia de 96.5% neste conjunto de dados (Parkhi; Vedaldi; Zisserman, 2015).

- MS-Celeb-1M: Este é um dos maiores conjuntos de dados de reconhecimento facial, onde modelos com o ArcFace alcançam acurácias superiores a 99% (Deng *et al.*, 2019).

A acurácia dos modelos treinados no FairFace apresentam acurácias que variam dependendo da complexidade e da arquitetura do modelo. Trabalhos que utilizaram redes neurais avançadas, como a ResNet-50 ou variantes de FaceNet, relataram acurácias entre 95% e 98% (Karkkainen; Joo, 2021).

Portanto, como sucesso deste trabalho devemos considerar a acurácia entre 95% a 98%, dentro dos grupos geográficos estudados.

4 AVALIAÇÃO EXPERIMENTAL

A avaliação experimental se divide em duas etapas, sendo que na primeira etapa, foi realizada uma análise detalhada do conjunto de dados *FairFace*, abrangendo a análise estatística, a verificação de valores nulos e a distribuição das classes. Já na segunda etapa, foram aplicados os algorítimos de treinamento no conjunto de dados, realizando variações nos hiper-parâmetros e analisando os resultados.

4.1 Seleção e Análise do Conjunto de Dados

Tamanho Conjunto de Dados

A tabela 1 representa o tamanho do conjunto de dados utilizado nos experimentos.

Tabela 1 – Tamanho Conjunto de Dados

Linhas	**Colunas**
97698	5

Resumo Estatístico

A tabela 2 apresenta um resumo estatístico do conjunto de dados utilizado nos experimentos.

Tabela 2 – Resumo dos Dados

	file	**age**	**gender**	**race**	**service_test**
count	97698	97698	97698	97698	97698
unique	97698	9	2	7	2
top	train/1.jpg	20-29	Male	White	False
freq	1	28898	51778	18612	52284

Verificando Valores Nulos

A tabela 3 apresenta a análise de valores nulos por coluna do conjunto de dados utilizado nos experimentos.

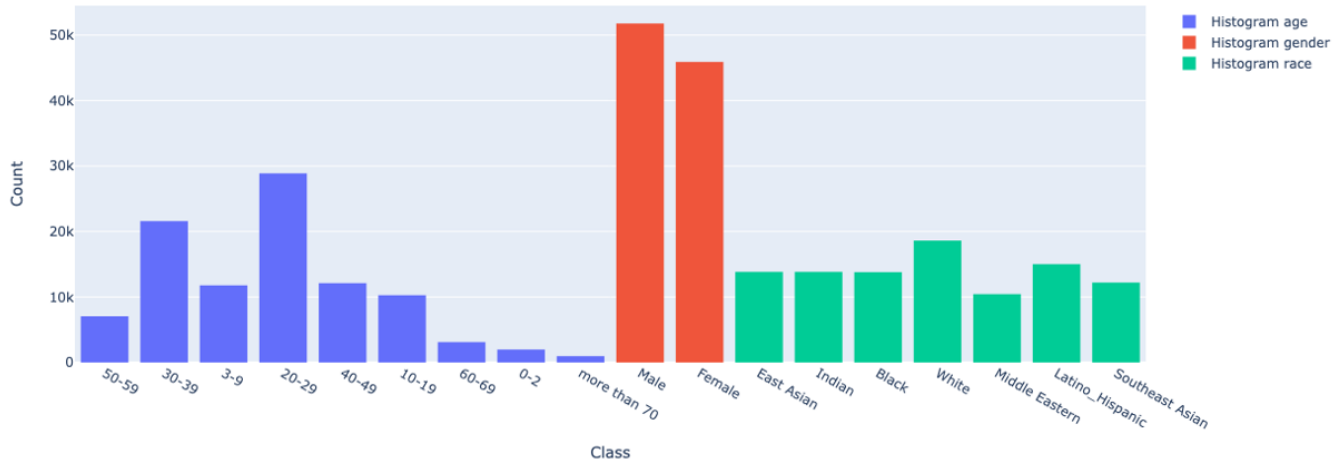
A Figura 15 apresenta um *dashboard* analítico das classes presentes no conjunto de dados original.

Com o objetivo de mitigar viés no reconhecimento de faces, a classe *race* foi selecionada para análise. A partir da análise dessa classe, identificou-se uma discrepância amostral. Portanto, foi aplicado o balanceamento utilizando a técnica de *undersampling* na menor classe.

Tabela 3 – Valores Nulos por Coluna

Colunas	Valores Nulos
file	0
age	0
gender	0
race	0
service_test	0

Figura 15 – Analítico das Classes Presentes no Conjunto de Dados Original



Fonte: Próprio Autor

Quantidade de Amostras Por Classe Antes do Balanceamento

A tabela 4 apresenta a quantidade de amostrar por classe antes do processo de balanceamento do conjunto de dados utilizado nos experimentos.

Tabela 4 – Contagem de Amostrar Antes do Balanceamento

Race	Count
White	18612
Latino Hispanic	14990
East Asian	13837
Indian	13835
Black	13789
Southeast Asian	12210
Middle Eastern	10425

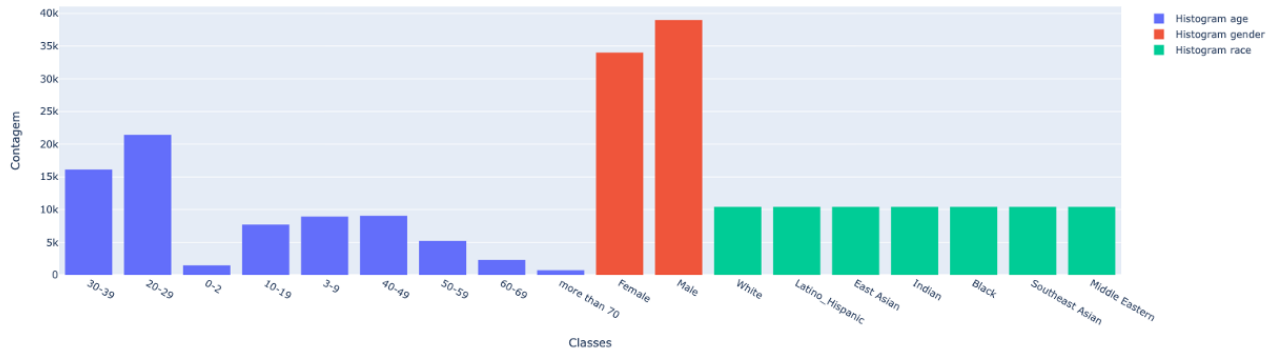
Quantidade de Amostras Por Classe Depois do Balanceamento

A tabela 5 apresenta a quantidade de amostrar por classe antes do processo de balanceamento do conjunto de dados utilizado nos experimentos.

Tabela 5 – Contagem de Amostrar Após o Balanceamento

Race	Count
White	10425
Latino Hispanic	10425
East Asian	10425
Indian	10425
Black	10425
Southeast Asian	10425
Middle Eastern	10425

Figura 16 – Analítico das Classes Presentes no Conjunto de Dados Balanceado



Fonte: Próprio Autor

A Figura 16 mostra o *dashboard* analítico das classes no conjunto de dados balanceado.

Amostras da Classe *Race*

A Figura 17 traz uma amostra de cada uma das 7 raças presentes no conjunto de dados balanceado, destacando a proximidade visual dos tons de pele e características faciais das classes Southeast Asian, East Asian, Indian e Latino Hispanic.

4.2 Pré-Processamento dos Dados

Realizaram-se dois experimentos para preparar as imagens para o treinamento do modelo.

Experimento I - Rotações e Detecção com MTCNN: Imagens foram rotacionadas em 45 graus utilizando a biblioteca *OpenCV*. Foram analisadas 20 imagens com 8 ângulos diferentes, totalizando 160 imagens. O algoritmo MTCNN detectou 102 faces, resultando em uma taxa de detecção de 63%. A Figura 18 ilustra uma amostra das imagens rotacionadas. A Figura 19 mostra alguns exemplos do desempenho do MTCNN na detecção de faces em imagens rotacionadas.

Figura 17 – Amostra das Raças do Conjunto de Dados

Fonte: Próprio Autor

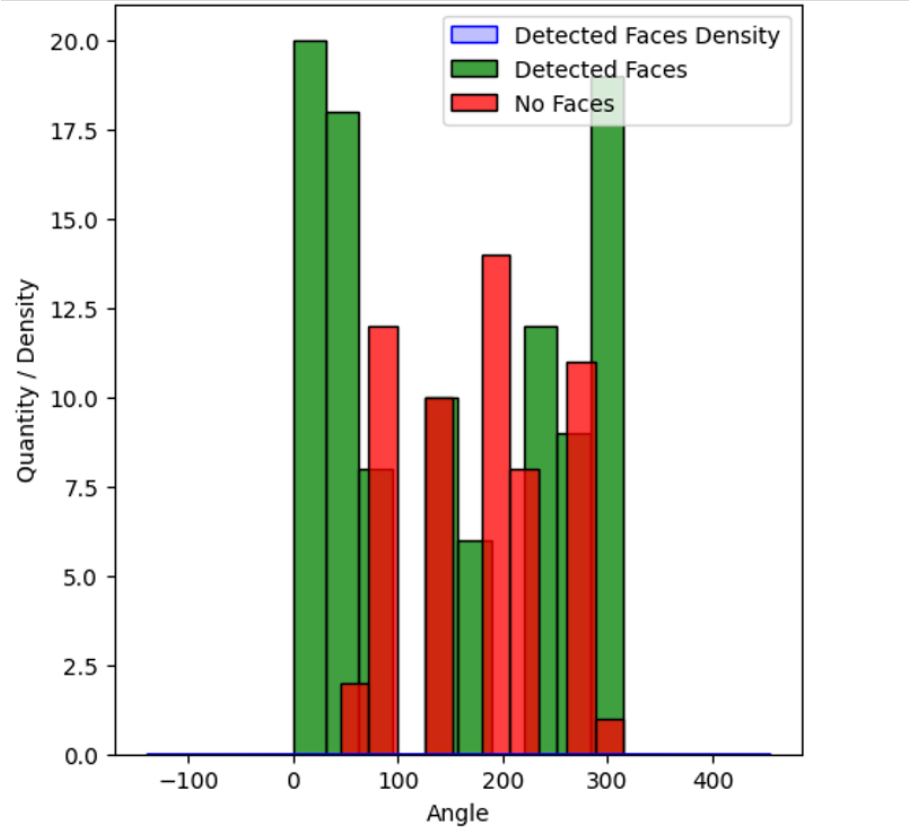
Figura 18 – Exemplo de Uma Imagem Rotacionada em 45 Graus

Fonte: Próprio Autor

Figura 19 – Exemplo de 3 Imagens Detectadas pelo MTCNN

Fonte: Próprio Autor

Figura 20 – Densidade das Faces Detectadas pelo Angulo Rotacionado

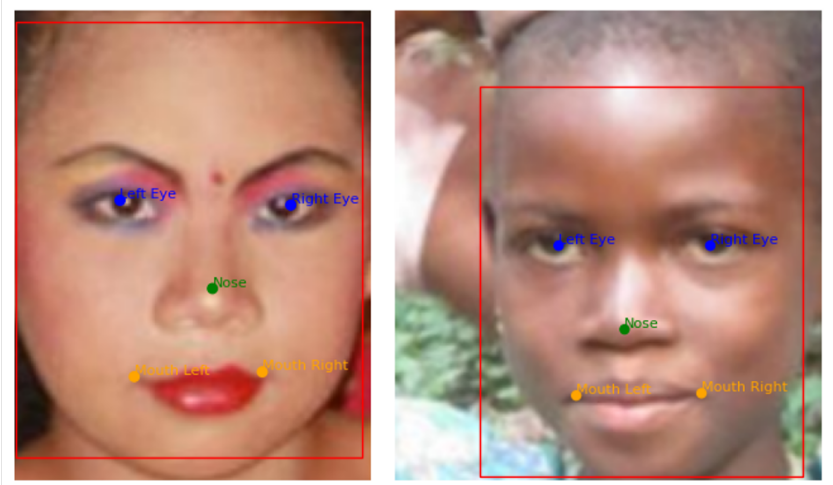


Fonte: Próprio Autor

A análise mostra que a taxa de falha é maior para ângulos próximos a 90 e 180 graus, com maior sucesso nos ângulos de 0 e 360 graus, conforme demonstrado na Figura 20. Também foi observada uma menor taxa de detecção para faces da classe *Black*.

Experimento II - Pós-Tratamento com MTCNN: As imagens que passaram por rotação, alinhamento e redimensionamento foram novamente processadas pelo MTCNN para assegurar a qualidade e assertividade da detecção. A Figura 21 exibe os cinco marcos faciais de duas imagens amostrais. A análise confirmou que não houve perda de qualidade ou assertividade após o pré-processamento.

Figura 21 – Exemplo dos Cinco Marcos Faciais de Duas Imagens Amostrais



Fonte: Próprio Autor

O pré-processamento incluiu a identificação de faces com MTCNN, o recorte, o alinhamento e o redimensionamento para 224 × 224 pixels. As imagens foram normalizadas para o intervalo [-1.0, 1.0] nos conjuntos de treinamento e validação.

4.3 Treinamento do Modelo

O modelo de rede neural convolucional (CNN) foi treinado utilizando a variante *LResNet50E-IR*, uma arquitetura baseada em ResNet-50 que incorpora melhorias para reconhecimento facial, como maior capacidade de extração de características discriminativas. No treinamento foram conduzidos alguns experimentos, com o objetivo de avaliar o desempenho do modelo sob diferentes configurações de função de perda e otimização:

- **Experimento 1**: Utilização da função de perda CrossEntropyLoss combinada com o otimizador SGD, utilizando o conjunto de dados balanceado e o *scheduler* OneCycleLR.

- **Experimento 2**: Utilização da função de perda ArcFaceLoss combinada com o otimizador SGD, utilizando o conjunto de dados balanceado e o *scheduler* OneCycleLR.

- **Experimento 3**: Utilização da função de perda CrossEntropyLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado e o *scheduler* OneCycleLR.

- **Experimento 4**: Utilização da função de perda ArcFaceLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado e o *scheduler* OneCycleLR.

- **Experimento 5**: Utilização da função de perda CrossEntropyLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado e o *scheduler* CosineAnnealingWarmRestarts.
- **Experimento 6**: Utilização da função de perda ArcFaceLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado e o *scheduler* CosineAnnealingWarmRestarts.
- **Experimento 7**: Utilização da função de perda CrossEntropyLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado, filtrado pelas classes alvo *"White"* e *"Black"* e o *scheduler* OneCycleLR.
- **Experimento 8**: Utilização da função de perda ArcFaceLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado, filtrado pelas classes alvo *"White"* e *"Black"* e o *scheduler* OneCycleLR.
- **Experimento 9**: Utilização da função de perda CrossEntropyLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado e o *scheduler* OneCycleLR, ainda com a mudança do *Dropout* para p=0.5.
- **Experimento 10**: Utilização da função de perda ArcFaceLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado e o *scheduler* OneCycleLR, ainda com a mudança do *Dropout* para p=0.5.
- **Experimento 11**: Utilização da função de perda CrossEntropyLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado e o *scheduler* OneCycleLR, considerando 40 épocas.

Para o otimizador SGD, foi adotado um valor de *momentum* de 0.9, que acelera o processo de convergência em direções relevantes ao suavizar oscilações. Além disso, foi utilizado um decaimento de peso (*weight decay*) de 0.0005, um fator que atua como regularizador, prevenindo sobre-ajuste ao controlar a magnitude dos pesos durante o treinamento.

Para o otimizador AdamW, foi utilizado um valor de *weight decay* de 0.0005, que atua como regularizador, prevenindo o sobre-ajuste ao penalizar grandes valores de pesos durante o treinamento. O AdamW combina as vantagens do método de descida de gradiente adaptativo, ajustando a taxa de aprendizado para cada parâmetro com base em estimativas de momentos de primeira e segunda ordem, e o decaimento de peso de maneira mais eficaz. Essa abordagem ajuda a melhorar a generalização do modelo, tornando-o menos suscetível ao sobre ajuste em dados complexos.

Para o *scheduler* OneCycleLR, foi adotada uma abordagem em que a taxa de aprendizado começa em um valor baixo, aumenta rapidamente até um valor máximo de

0.01, e depois decresce gradualmente até o final do treinamento. Esse ciclo único acelera a convergência inicial ao permitir uma exploração mais ampla do espaço de parâmetros, evitando que o modelo fique preso em mínimos locais. Ao mesmo tempo, a redução da taxa de aprendizado ao longo das últimas iterações melhora a estabilidade do modelo, garantindo que ele refine seus pesos de forma mais precisa. Esse comportamento cíclico otimiza o uso da taxa de aprendizado, resultando em uma convergência mais rápida e eficaz.

Para o *scheduler* CosineAnnealingWarmRestarts, foi adotada uma estratégia em que a taxa de aprendizado segue uma função cosseno decrescente ao longo de cada ciclo, com reinicializações periódicas de 8 épocas para um valor mais alto. Essas reinicializações, conhecidas como *warm restarts*, permitem que o modelo escape de mínimos locais, dando ao treinamento uma nova oportunidade de explorar soluções melhores. A taxa de aprendizado decresce suavemente até um valor mínimo, o que ajuda na estabilidade do treinamento, enquanto os *restarts* permitem uma recuperação eficiente de regiões promissoras no espaço de parâmetros.

O conjunto de dados foi dividido em três subconjuntos: 80% dos dados foram destinados ao treinamento, 10% para validação, e os 10% restantes foram reservados para testes finais. A divisão foi feita de forma estratificada, garantindo que a distribuição das classes fosse balanceada em cada um dos subconjuntos.

O *batch size* escolhido para ambos os experimentos foi de 128, uma configuração que equilibra o uso eficiente da memória da GPU e a estabilidade do gradiente durante a otimização. A escolha desse tamanho também leva em consideração a capacidade da infraestrutura utilizada.

4.3.1 Ambiente de Treinamento

O ambiente de treinamento foi configurado utilizando recursos de computação em nuvem da Microsoft Azure, devido à sua escalabilidade e suporte avançado para *workloads* intensivas em GPU, como aquelas utilizadas em modelos de aprendizado profundo (Azure, 2024). A máquina virtual selecionada para os experimentos foi configurada com as seguintes especificações:

- **Tamanho da máquina virtual**: *Standard_NC6s_v3*, que oferece 6 núcleos de CPU, 112 GB de RAM e um disco de 736 GB. Este tipo de máquina é otimizado para workloads que exigem alto desempenho computacional, particularmente para operações de treinamento de modelos com GPU.

- **Unidade de processamento gráfico (GPU)**: 1 NVIDIA Tesla V100.

Além dos recursos computacionais, o conjunto de dados original utilizado foi armazenado em um bucket do serviço de armazenamento de objetos da Amazon Web Services (AWS), o Amazon S3. Essa solução foi escolhida pela sua confiabilidade e alta disponibilidade, garantindo acessibilidade e velocidade no carregamento dos dados (Services, 2024).

A utilização de recursos de nuvem para esse projeto se justifica pela flexibilidade na configuração de máquinas virtuais e pelo suporte ao uso de hardware especializado, como GPUs, essenciais para o treinamento eficiente de redes neurais profundas. Esses ambientes são amplamente utilizados na comunidade acadêmica e na indústria devido ao seu suporte robusto e escalável para pesquisa e desenvolvimento em inteligência artificial.

Abaixo são apresentadas as versões dos aplicativos, softwares e principais bibliotecas utilizadas:

- Sistema Operacional: Linux 5.15.0-1064-azure
- Processor: x86_64
- Python Version: 3.9.19
- PyTorch Version: 2.4.1
- Numpy Version: 1.23.5
- Pandas Version: 1.3.5
- Torchvision Version: 0.14.1

4.3.2 Classe LResNet50E-IR

Essa classe define a arquitetura do modelo LResNet50E_IR. O código desenvolvido utiliza a ResNet50 pré-treinada do *torchvision*, alterando a última camada para se adequar ao número de classes do conjunto de dados, através da camada fc da ResNet50. O self.fc é a nova camada final que será usada para mapear os vetores para as classes. Um componente adicional de *Dropout* com p=0.2 foi utilizado para regularizar o treinamento, reduzindo *overfitting*.

4.3.3 Acesso ao Código Fonte

Todo o código fonte está disponível no GitHub (Ozzetti, 2024).

4.4 Análise de Desempenho do Modelo Treinado

Abaixo são apresentados os testes realizados, bem como os resultados apurados, considerando a variação das funções de perda, dos otimizadores e dos *schedullers*.

Os gráficos das métricas de treinamento possuem 4 quadrantes, onde o primeiro quadrante apresenta o gráfico de *"Loss over Epochs"* que representa o erro médio do modelo após cada época de treinamento. Uma diminuição constante indica que o modelo está se ajustando aos dados. O segundo quadrante apresenta o gráfico de *"Accuracy over Epochs"* que mede a porcentagem de previsões corretas, e um aumento ao longo do tempo sugere que o modelo está aprendendo a classificar corretamente as amostras. No terceiro quadrante apresenta o gráfico *"Precision over Epochs"* que foca nas previsões corretas de uma classe específica em relação ao total de previsões feitas para essa classe, sendo útil para problemas onde classes desbalanceadas ou erros de classificação têm diferentes impactos. Por fim, o ultimo quadrante apresenta o gráfico da *"Log Loss over Epochs"* que penaliza previsões de baixa confiança mesmo quando corretas, avaliando o quanto o modelo é capaz de atribuir probabilidades corretas às classes.

Para cada experimento também será explorada a Matriz de Confusão, que representa uma ferramenta essencial na avaliação de modelos de classificação, especialmente em Inteligência Artificial, fornecendo uma visão detalhada do desempenho do modelo ao comparar previsões com valores reais. Organizada em uma matriz quadrada, ela apresenta quatro métricas principais: Verdadeiros Positivos (VP), Falsos Positivos (FP), Verdadeiros Negativos (VN) e Falsos Negativos (FN). Esses valores ajudam a compreender onde o modelo acerta e onde comete erros, permitindo o cálculo de métricas como acurácia, precisão, *recall* e *F1-score*. Em contextos práticos, a análise da Matriz de Confusão é crucial para identificar classes que o modelo pode ter dificuldades em distinguir, auxiliando na calibração do modelo e no aprimoramento de sua performance em problemas específicos de classificação.

4.4.1 Resultado Experimento 1

Neste experimento foi utilizada a função de perda CrossEntropyLoss combinada com o otimizador SGD, utilizando o conjunto de dados balanceado e o *scheduler* OneCycleLR.

4.4.1.1 Métricas de Treinamento

Na Figura 22 são apresentadas as métricas do treinamento.

Pode ser

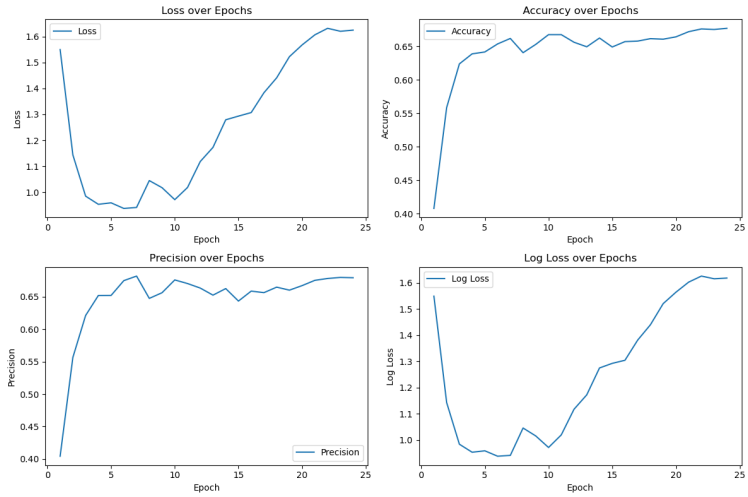
4.4.1.2 Matriz de Confusão

Na Figura 23 é apresentada a matriz de confusão do treinamento.

4.4.1.3 Relatório de Classificação

Na tabela 6 é apresentado o relatório de classificação.

Figura 22 – Métricas do Experimento 1



Fonte: Próprio Autor

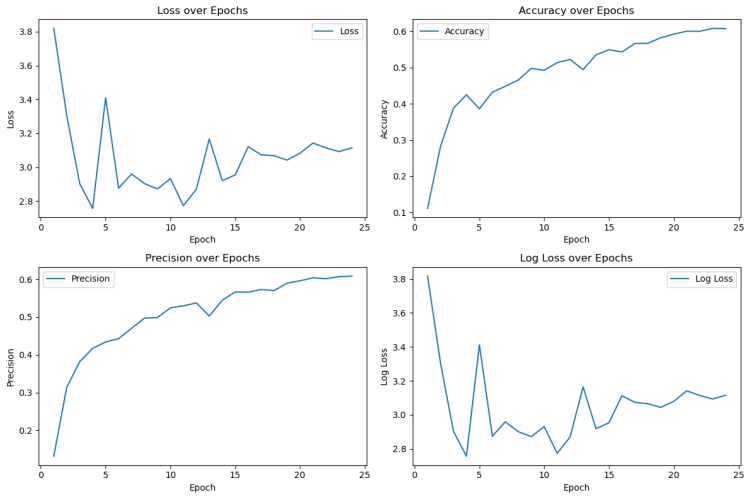
Figura 23 – Matriz de Confusão do Experimento 1

Fonte: Próprio Autor

Classe	Precisão	Recall	F1-Score	Support
Black	0.85	0.86	0.86	1005
s East Asian	0.71	0.72	0.71	1005
Indian	0.73	0.72	0.72	1005
Latino Hispanic	0.50	0.52	0.51	1005
Middle Eastern	0.68	0.70	0.69	1005
Southeast Asian	0.60	0.60	0.60	1005
White	0.71	0.64	0.68	1006
Acurácia	0.68			
Macro Avg	0.68	0.68	0.68	7036
Weighted Avg	0.68	0.68	0.68	7036

Tabela 6 – Relatório de Métricas de *precision*, *recall*, *F1-score*, e *support* do Experimento 1

Figura 24 – Métricas do Experimento 2



Fonte: Próprio Autor

4.4.2 Resultado Experimento 2

Neste experimento foi utilizada a função de perda ArcFaceLoss combinada com o otimizador SGD, utilizando o conjunto de dados balanceado e o *scheduler* OneCycleLR.

4.4.2.1 Métricas de Treinamento

Na Figura 24 são apresentadas as métricas do treinamento.

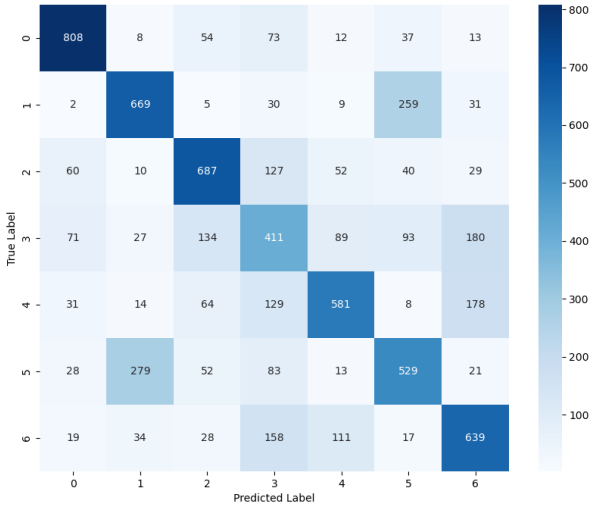
4.4.2.2 Matriz de Confusão

Na Figura 25 é apresentada a matriz de confusão do treinamento.

4.4.2.3 Relatório de Classificação

Na tabela 7 é apresentado o relatório de classificação.

Figura 25 – Matriz de Confusão do Experimento 2



Fonte: Próprio Autor

Classe	**Precisão**	**Recall**	**F1-Score**	**Support**
Black	0.79	0.80	0.80	1005
East Asian	0.64	0.67	0.65	1005
Indian	0.67	0.68	0.68	1005
Latino Hispanic	0.41	0.41	0.41	1005
Middle Eastern	0.67	0.58	0.62	1005
Southeast Asian	0.54	0.53	0.53	1005
White	0.59	0.64	0.61	1006
Acurácia	0.61			
Macro Avg	0.62	0.61	0.61	7036
Weighted Avg	0.62	0.61	0.61	7036

Tabela 7 – Relatório de Métricas de *precision*, *recall*, *F1-score*, e *support* do Experimento 2

4.4.3 Resultado Experimento 3

Neste experimento foi utilizada a função de perda CrossEntropyLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado e o *scheduler* OneCycleLR.

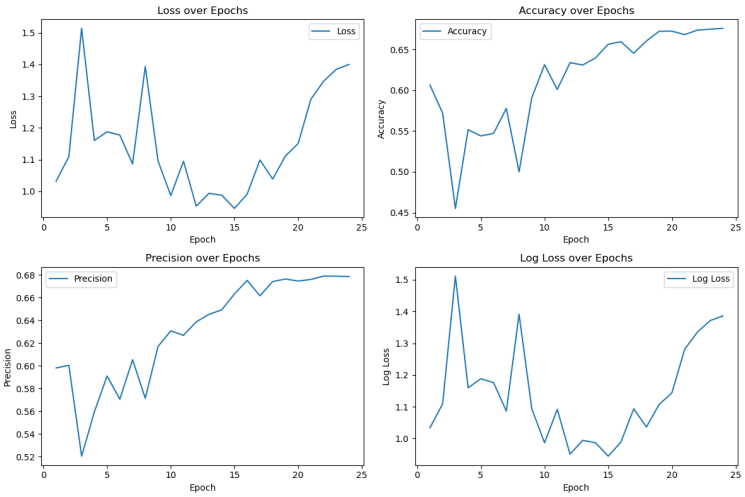
4.4.3.1 Métricas de Treinamento

Na Figura 26 são apresentadas as métricas do treinamento.

4.4.3.2 Matriz de Confusão

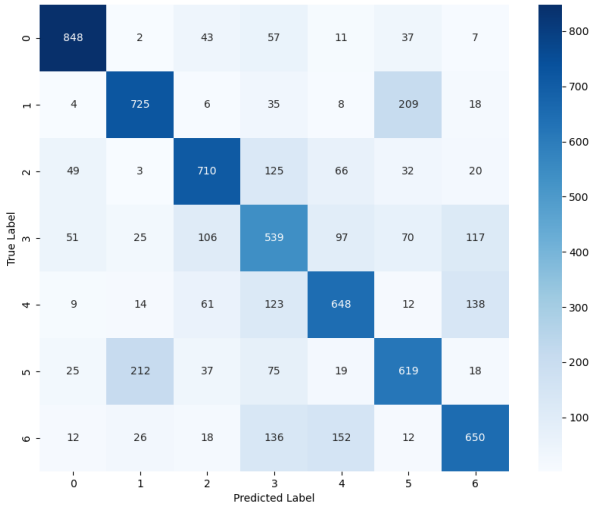
Na Figura 27 é apresentada a matriz de confusão do treinamento.

Figura 26 – Métricas do Experimento 3



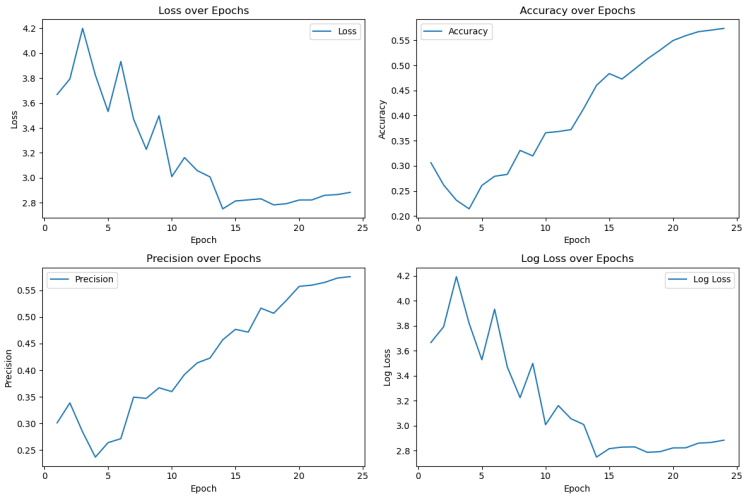
Fonte: Próprio Autor

Figura 27 – Matriz de Confusão do Experimento 3



Fonte: Próprio Autor

Figura 28 – Métricas do Experimento 4



Fonte: Próprio Autor

4.4.3.3 Relatório de Classificação

Na tabela 8 é apresentado o relatório de classificação.

Classe	**Precisão**	**Recall**	**F1-Score**	**Support**
Black	0.85	0.84	0.85	1005
East Asian	0.72	0.72	0.72	1005
Indian	0.72	0.71	0.72	1005
Latino Hispanic	0.49	0.54	0.51	1005
Middle Eastern	0.65	0.64	0.65	1005
Southeast Asian	0.62	0.62	0.62	1005
White	0.67	0.65	0.66	1006
Acurácia	0.67			
Macro Avg	0.68	0.67	0.67	7036
Weighted Avg	0.68	0.67	0.67	7036

Tabela 8 – Relatório de Métricas de *precision*, *recall*, *F1-score*, e *support* do Experimento 3

4.4.4 Resultado Experimento 4

Neste experimento foi utilizada a função de perda ArcFaceLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado e o *scheduler* OneCycleLR.

4.4.4.1 Métricas de Treinamento

Na Figura 28 são apresentadas as métricas do treinamento.

4.4.4.2 Matriz de Confusão

Na Figura 29 é apresentada a matriz de confusão do treinamento.

Figura 29 – Matriz de Confusão do Experimento 4

Fonte: Próprio Autor

4.4.4.3 Relatório de Classificação

Na tabela 9 é apresentado o relatório de classificação.

Classe	**Precisão**	**Recall**	**F1-Score**	**Support**
Black	0.84	0.79	0.81	1005
East Asian	0.62	0.68	0.65	1005
Indian	0.64	0.65	0.65	1005
Latino Hispanic	0.34	0.36	0.35	1005
Middle Eastern	0.54	0.55	0.55	1005
Southeast Asian	0.53	0.48	0.50	1005
White	0.59	0.57	0.58	1006
Acurácia	0.58			
Macro Avg	0.58	0.58	0.58	7036
Weighted Avg	0.58	0.58	0.58	7036

Tabela 9 – Relatório de Métricas de *precision*, *recall*, *F1-score*, e *support* do Experimento 4

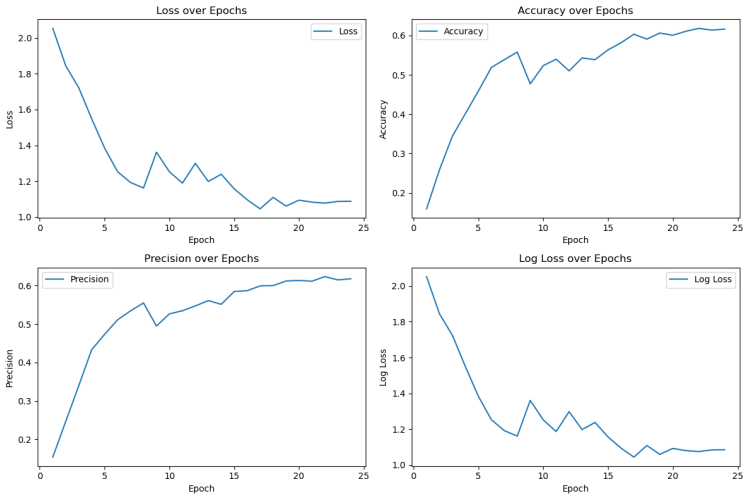
4.4.5 Resultado Experimento 5

Neste experimento foi utilizada a função de perda CrossEntropyLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado e o *scheduler* CosineAnnealingWarmRestarts.

4.4.5.1 Métricas de Treinamento

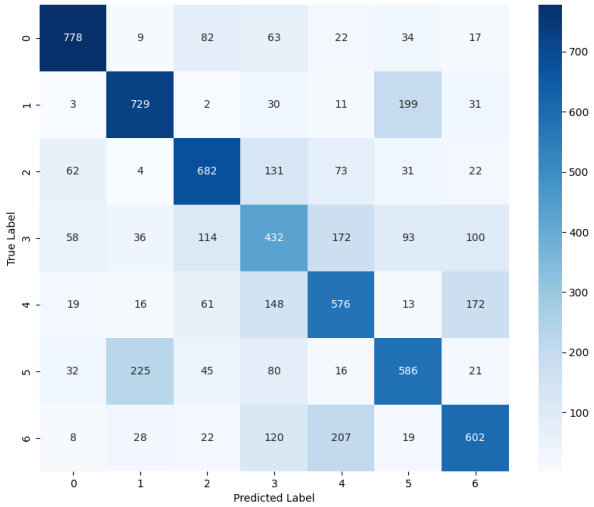
Na Figura 30 são apresentadas as métricas do treinamento.

Figura 30 – Métricas do Experimento 5



Fonte: Próprio Autor

Figura 31 – Matriz de Confusão do Experimento 5



Fonte: Próprio Autor

4.4.5.2 Matriz de Confusão

Na Figura 31 é apresentada a matriz de confusão do treinamento.

4.4.5.3 Relatório de Classificação

Na tabela 10 é apresentado o relatório de classificação.

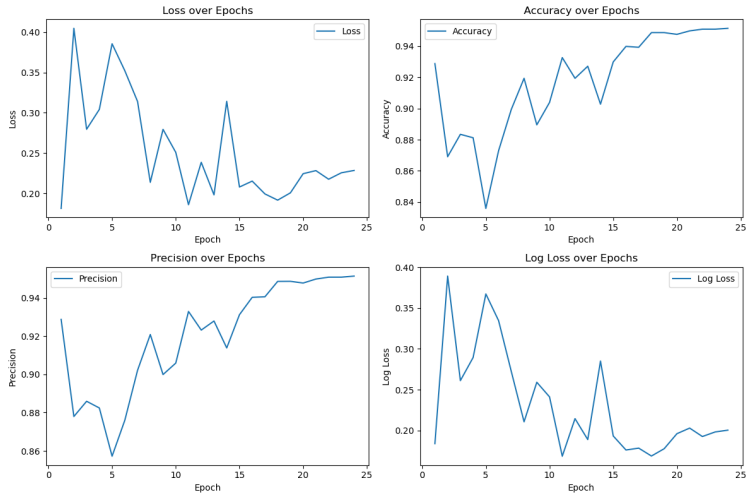
4.4.6 Resultado Experimento 6

Neste experimento foi utilizada a função de perda ArcFaceLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado e o *scheduler* CosineAnne-

Classe	Precisão	Recall	F1-Score	Support
Black	0.81	0.77	0.79	1005
East Asian	0.70	0.73	0.71	1005
Indian	0.68	0.68	0.68	1005
Latino Hispanic	0.43	0.43	0.43	1005
Middle Eastern	0.53	0.57	0.55	1005
Southeast Asian	0.60	0.58	0.59	1005
White	0.62	0.60	0.61	1006
Acurácia	0.62			
Macro Avg	0.62	0.62	0.62	7036
Weighted Avg	0.62	0.62	0.62	7036

Tabela 10 – Relatório de Métricas de *precision*, *recall*, *F1-score*, e *support* do Experimento 5

Figura 32 – Métricas do Experimento 7



Fonte: Próprio Autor

alingWarmRestarts.

Neste experimento foi desconsiderada a coleta dos dados de resultados devido a apresentação das métricas zeradas.

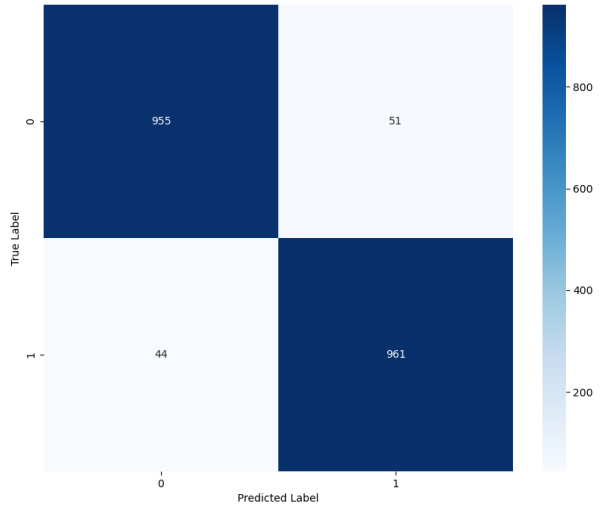
4.4.7 Resultado Experimento 7

Neste experimento foi utilizada a função de perda CrossEntropyLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado, filtrado pelas classes alvo *"White"* e *"Black"* e o *scheduler* OneCycleLR.

4.4.7.1 Métricas de Treinamento

Na Figura 32 são apresentadas as métricas do treinamento.

Figura 33 – Matriz de Confusão do Experimento 7



Fonte: Próprio Autor

4.4.7.2 Matriz de Confusão

Na Figura 33 é apresentada a matriz de confusão do treinamento.

4.4.7.3 Relatório de Classificação

Na tabela 11 é apresentado o relatório de classificação.

Classe	**Precisão**	**Recall**	**F1-Score**	**Support**
Black	0.96	0.95	0.95	1006
White	0.95	0.96	0.95	1005
Acurácia	0.95			
Macro Avg	0.95	0.95	0.95	2011
Weighted Avg	0.95	0.95	0.95	2011

Tabela 11 – Relatório de Métricas de *precision*, *recall*, *F1-score*, e *support* do Experimento 7

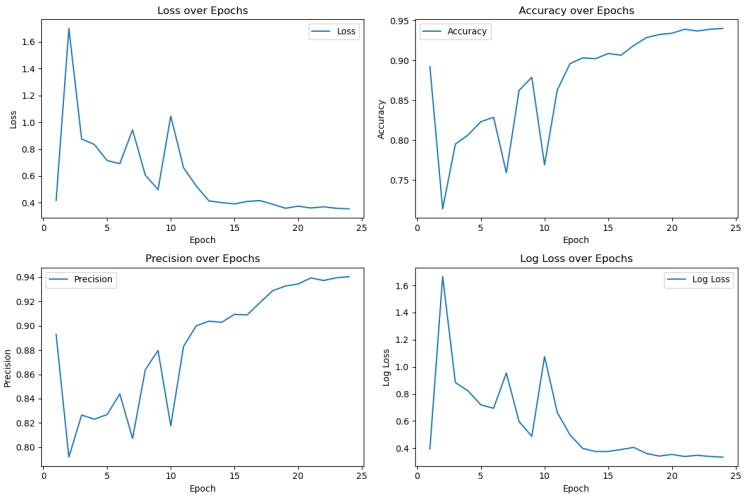
4.4.8 Resultado Experimento 8

Neste experimento foi utilizada a função de perda ArcFaceLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado, filtrado pelas classes alvo *"White"* e *"Black"* e o *scheduler* OneCycleLR.

4.4.8.1 Métricas de Treinamento

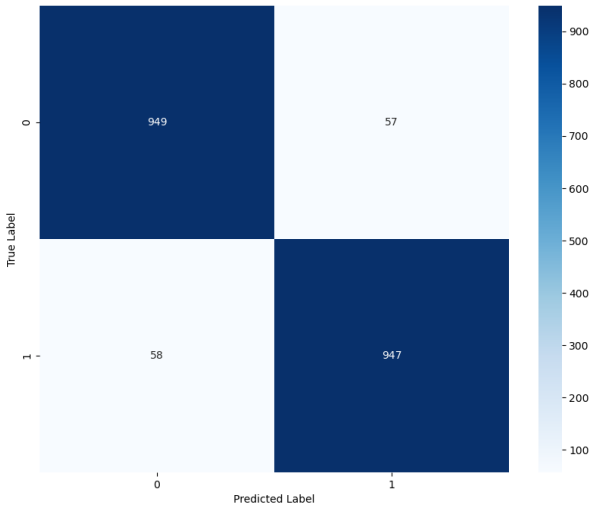
Na Figura 34 são apresentadas as métricas do treinamento.

Figura 34 – Métricas do Experimento 8



Fonte: Próprio Autor

Figura 35 – Matriz de Confusão do Experimento 8



Fonte: Próprio Autor

4.4.8.2 Matriz de Confusão

Na Figura 35 é apresentada a matriz de confusão do treinamento.

4.4.8.3 Relatório de Classificação

Na tabela 12 é apresentado o relatório de classificação.

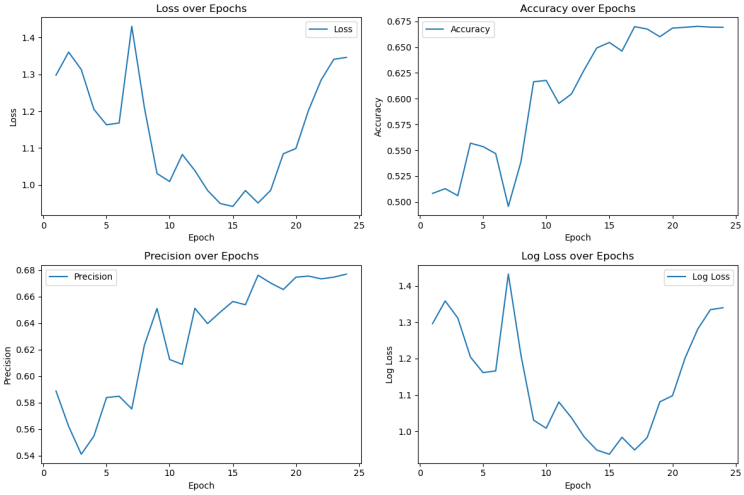
4.4.9 Resultado Experimento 9

Neste experimento foi utilizada a função de perda CrossEntropyLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado e o *scheduler*

Classe	Precisão	Recall	F1-Score	Support
Black	0.94	0.94	0.94	1006
White	0.94	0.94	0.94	1005
Acurácia	0.94			
Macro Avg	0.94	0.94	0.94	2011
Weighted Avg	0.94	0.94	0.94	2011

Tabela 12 – Relatório de Métricas de *precision*, *recall*, *F1-score*, e *support* do Experimento 8

Figura 36 – Métricas do Experimento 9



Fonte: Próprio Autor

OneCycleLR, ainda com a mudança do *Dropout* para p=0.5.

4.4.9.1 Métricas de Treinamento

Na Figura 36 são apresentadas as métricas do treinamento.

4.4.9.2 Matriz de Confusão

Na Figura 37 é apresentada a matriz de confusão do treinamento.

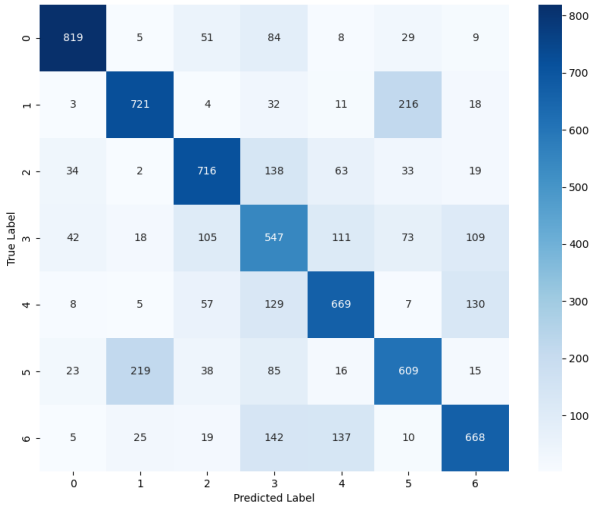
4.4.9.3 Relatório de Classificação

Na tabela 13 é apresentado o relatório de classificação.

4.4.10 Resultado Experimento 10

Neste experimento foi utilizada a função de perda ArcFaceLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado e o *scheduler* OneCycleLR, ainda com a mudança do *Dropout* para p=0.5.

Figura 37 – Matriz de Confusão do Experimento 9



Fonte: Próprio Autor

Classe	**Precisão**	**Recall**	**F1-Score**	**Support**
Black	0.88	0.81	0.84	1005
East Asian	0.72	0.72	0.72	1005
Indian	0.72	0.71	0.72	1005
Latino Hispanic	0.47	0.54	0.51	1005
Middle Eastern	0.66	0.67	0.66	1005
Southeast Asian	0.62	0.61	0.61	1005
White	0.69	0.66	0.68	1006
Acurácia		0.67		
Macro Avg	0.68	0.67	0.68	7036
Weighted Avg	0.68	0.67	0.68	7036

Tabela 13 – Relatório de Métricas de *precision*, *recall*, *F1-score*, e *support* do Experimento 9

4.4.10.1 Métricas de Treinamento

Na Figura 38 são apresentadas as métricas do treinamento.

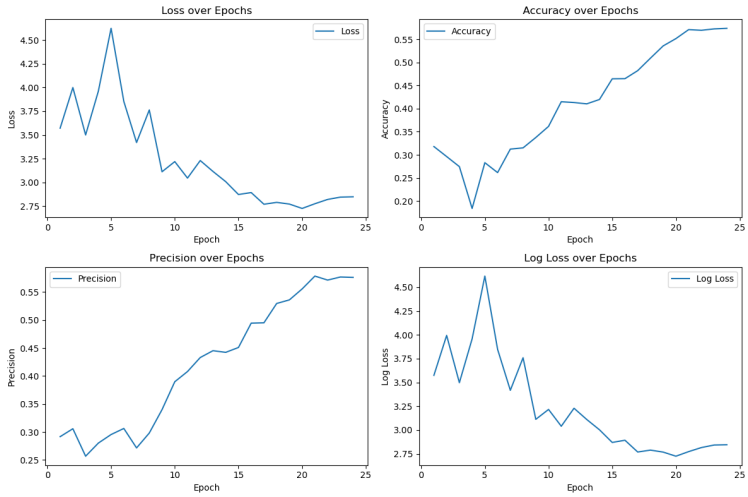
4.4.10.2 Matriz de Confusão

Na Figura 39 é apresentada a matriz de confusão do treinamento.

4.4.10.3 Relatório de Classificação

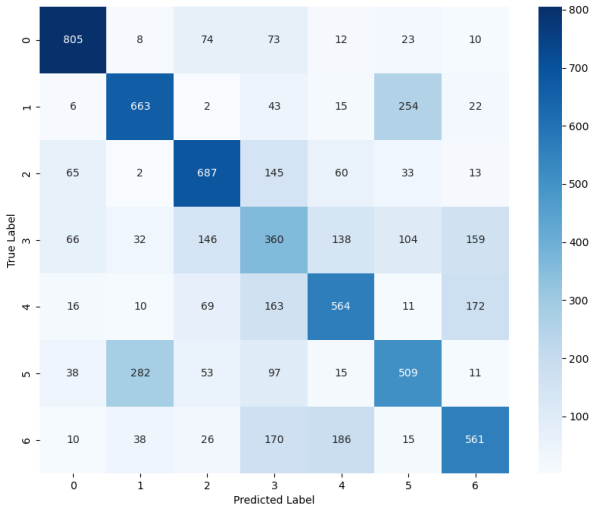
Na tabela 14 é apresentado o relatório de classificação.

Figura 38 – Métricas do Experimento 10



Fonte: Próprio Autor

Figura 39 – Matriz de Confusão do Experimento 10

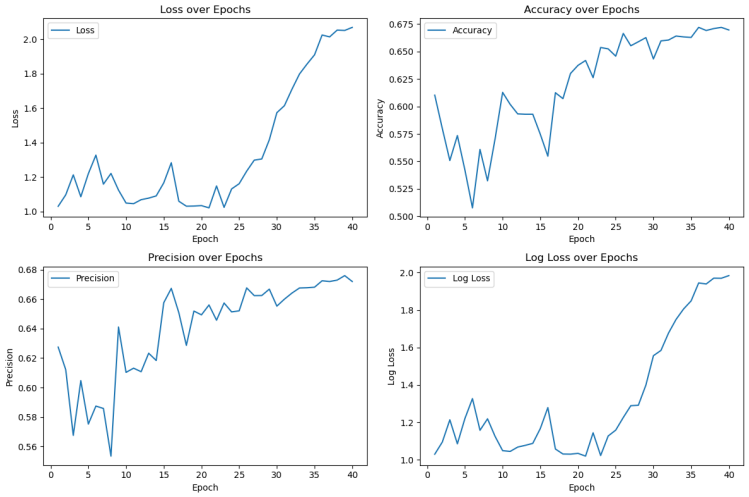


Fonte: Próprio Autor

Classe	Precisão	Recall	F1-Score	Support
Black	0.80	0.80	0.80	1005
East Asian	0.64	0.66	0.65	1005
Indian	0.65	0.68	0.67	1005
Latino Hispanic	0.34	0.36	0.35	1005
Middle Eastern	0.57	0.56	0.57	1005
Southeast Asian	0.54	0.51	0.52	1005
White	0.59	0.56	0.57	1006
Acurácia	0.59			
Macro Avg	0.59	0.59	0.59	7036
Weighted Avg	0.59	0.59	0.59	7036

Tabela 14 – Relatório de Métricas de *precision*, *recall*, *F1-score*, e *support* do Experimento 10

Figura 40 – Métricas do Experimento 11



Fonte: Próprio Autor

4.4.11 Resultado Experimento 11

Neste experimento foi utilizada a função de perda CrossEntropyLoss combinada com o otimizador AdamW, utilizando o conjunto de dados balanceado e o *scheduler* OneCycleLR, considerando 40 épocas.

4.4.11.1 Métricas de Treinamento

Na Figura 40 são apresentadas as métricas do treinamento.

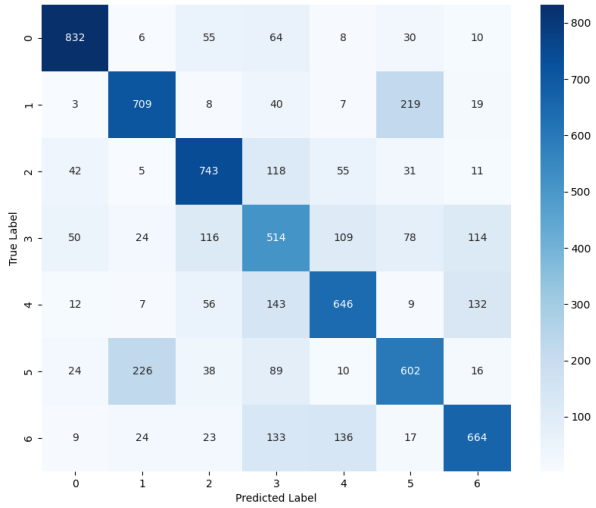
4.4.11.2 Matriz de Confusão

Na Figura 41 é apresentada a matriz de confusão do treinamento.

4.4.11.3 Relatório de Classificação

Na tabela 15 é apresentado o relatório de classificação.

Figura 41 – Matriz de Confusão do Experimento 11



Fonte: Próprio Autor

Tabela 15 – Relatório de Métricas de *precision*, *recall*, *F1-score*, e *support* do Experimento 11

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Black	0.86	0.83	0.84	1005
East Asian	0.71	0.71	0.71	1005
Indian	0.72	0.74	0.73	1005
Latino Hispanic	0.47	0.51	0.49	1005
Middle Eastern	0.67	0.64	0.65	1005
Southeast Asian	0.61	0.60	0.60	1005
White	0.69	0.66	0.67	1006
Accuracy			0.67	7036
Macro Avg	0.67	0.67	0.67	7036
Weighted Avg	0.67	0.67	0.67	7036

4.4.12 Análise dos Resultados

Nesta seção, analisamos os resultados dos experimentos de reconhecimento facial, utilizando diferentes funções de perda, otimizadores e *schedulers*. A análise é conduzida em termos das métricas de desempenho definidas e comparando-as com as diferentes combinações experimentadas.

4.4.12.1 Observações sobre os Experimentos

A seguir são apresentadas as observações sobre os experimentos realizados:

- O experimento 1 apresentou o melhor desempenho geral, com uma acurácia de 0.68. Essa combinação de CrossEntropyLoss e SGD demonstrou ser mais eficaz para o

conjunto de dados balanceado em comparação com ArcFaceLoss.

- O experimento 4, que também utilizou ArcFaceLoss com AdamW, mostrou a menor acurácia de 0.58. Isso pode indicar que a combinação de AdamW com ArcFaceLoss pode não ser tão adequada para este cenário, possivelmente devido a uma falta de convergência durante o treinamento.

- O experimento 5 apresentou uma acurácia de 0.62, levenemten menor dos melhores resultados dos experimentos anteriores, mas ainda assim representa um desempenho aceitável considerando a utilização do *scheduler* CosineAnnealingWarmRestarts.

- No experimento 6, a ausência de resultados relevantes (zerados) indica que a combinação de ArcFaceLoss com AdamW e o *scheduler* pode não ter funcionado adequadamente, possivelmente devido a problemas de convergência ou aprendizado inadequado.

- Ambos os experimentos 7 e 8 mostraram resultados promissores, com altas acurácias de 0.95 e 0.94, e f1-scores médios de 0.95 e 0.94, respectivamente. Isso indica que a filtragem das classes alvo melhorou significativamente o desempenho do modelo, permitindo uma classificação mais precisa entre grupos demográficos específicos.

- O experimento 9 manteve um desempenho razoável com uma acurácia de 0.67, enquanto o experimento 10 caiu para 0.59, refletindo a tendência observada nos experimentos anteriores com ArcFaceLoss.

- A mudança do *dropout* no experimento 10 indica que a combinação de ArcFaceLoss e AdamW, neste contexto, se manteve menos eficaz em comparação com CrossEntropyLoss.

Por fim, ao comparar o desempenho do Experimento 11 com os demais experimentos conduzidos, podemos observar algumas tendências e diferenças significativas:

- Acurácia Global: O experimento obteve uma acurácia geral de 67%, similar à de experimentos anteriores, como o Experimento 5 (62%) e o Experimento 7 (95%). Embora o Experimento 7 tenha apresentado uma acurácia significativamente maior, o Experimento 11 destaca-se por manter uma consistência em termos da média F1-Score (0.67), mostrando uma distribuição equilibrada do desempenho entre as classes.

- O aumento das épocas para 40, não refletiu ganhos significativos para o treinamento.

- Desempenho da Classe Black: O desempenho mais alto foi observado nesta classe, com um F1-Score de 0.84, o que reflete a boa capacidade do modelo em reconhecer

corretamente indivíduos dessa classe. Este resultado está entre os melhores observados nos experimentos realizados.

- Desempenho das Classes East Asian e Indian: Ambas as classes tiveram resultados razoáveis, com F1-Scores de 0.71 e 0.73, respectivamente. Estes valores são próximos aos observados nos experimentos 9 e 10, que também utilizaram AdamW.

- Desempenho da Classe Latino Hispanic: Esta classe apresentou o desempenho mais baixo, com um F1-Score de 0.49. A dificuldade em identificar indivíduos desta classe foi observada em experimentos anteriores e sugere que o modelo enfrenta desafios específicos para este grupo, possivelmente relacionados à variação de características faciais ou ao equilíbrio do conjunto de dados.

4.4.12.2 Comparação entre Funções de Perda

Os resultados obtidos a partir da aplicação de diferentes funções de perda mostram um desempenho variado em termos de acurácia e métricas de F1-score. Experimentos utilizando a função de perda CrossEntropyLoss (Experimentos 1, 3, 5, 7 e 9) apresentaram consistentemente melhores resultados em comparação à ArcFaceLoss (Experimentos 2, 4, 6, 8 e 10). A função CrossEntropyLoss alcançou um F1-score máximo de 0.91 no Experimento 9, evidenciando sua eficácia em classes mais representadas, enquanto a ArcFaceLoss apresentou um desempenho inferior, particularmente em classes com suporte mais reduzido, como "Latino Hispanic".

4.4.12.3 Impacto dos Otimizadores

A comparação dos resultados entre os otimizadores AdamW e SGD revela que o AdamW (Experimentos 3, 5, 7, 9) teve um impacto positivo nas métricas de desempenho, resultando em maiores acurácias e F1-scores. Essa diferença é notável especialmente em experimentos onde a taxa de aprendizado foi adequadamente ajustada, contribuindo para uma melhor convergência. Por outro lado, o SGD (Experimentos 1, 2, 4, 6, 8 e 10) apresentou resultados mais modestos, sugerindo que a escolha do otimizador é crítica para o sucesso do modelo.

4.4.12.4 Efeito do *Scheduler*

A análise dos *schedullers* de aprendizado revela que a implementação do OneCycleLR (Experimentos 3, 5, 7) favoreceu a convergência e melhorou as métricas de acurácia e F1-score, em comparação ao CosineAnnealingWarmRestarts (Experimentos 5 e 6). A variação da taxa de aprendizado proporcionada pelo OneCycleLR contribuiu para um treinamento mais eficaz e consistente.

4.4.12.5 Análise da Matriz de Confusão

A avaliação das matrizes de confusão indicou que a classe "Latino Hispanic"foi uma constante fonte de erros, apresentando baixos valores de precisão e *recall* em todos os experimentos. Essa observação sugere a necessidade de intervenções específicas, como técnicas de *data augmentation*, para equilibrar a representação das classes e melhorar o desempenho do modelo.

4.4.12.6 Matriz Geral dos Resultados dos Experimentos

A tabela 16 apresenta a visão geral dos resultados dos experimentos.

Tabela 16 – Resultados dos Experimentos

Experimento	**Função de Perda**	**Otimizador**	**Acurácia**	**F1-Score**	**Precisão**
1	CrossEntropyLoss	SGD	0.68	0.68	0.68
2	ArcFaceLoss	SGD	0.61	0.62	0.61
3	CrossEntropyLoss	AdamW	0.67	0.68	0.67
4	ArcFaceLoss	AdamW	0.58	0.58	0.58
5	CrossEntropyLoss	AdamW	0.62	0.62	0.62
6	ArcFaceLoss	AdamW	-	-	-
7	CrossEntropyLoss	AdamW	0.95	0.95	0.95
8	ArcFaceLoss	AdamW	0.94	0.94	0.94
9	CrossEntropyLoss	AdamW	0.67	0.67	0.67
10	ArcFaceLoss	AdamW	0.59	0.59	0.59
11	CrossEntropyLoss	AdamW	0.67	0.67	0.67

5 CONCLUSÕES

5.1 Síntese dos Pontos Abordados

Os experimentos realizados mostraram resultados consistentes, com acurácias variando entre 0.58 e 0.95, dependendo da técnica utilizada. A utilização do otimizador AdamW, em combinação com a função de perda CrossEntropyLoss, apresentou os melhores resultados, alcançando uma acurácia de até 0.95. Em contraste, a função de perda ArcFaceLoss, apesar de promissora para o reconhecimento de faces, apresentou um desempenho inferior nos experimentos conduzidos.

Ainda que a acurácia geral dos experimentos tenha variado entre 0.58 e 0.95, indicando um desempenho satisfatório, há espaço significativo para melhorias, especialmente em classes menos representadas. A continuidade na exploração de técnicas avançadas de *data augmentation* e outras abordagens de balanceamento de classes será essencial para aprimorar a eficácia do modelo.

Ainda, ao comparar os resultados dos experimentos que utilizaram diferentes otimizadores, foi possível observar que o uso do AdamW proporcionou melhorias significativas na acurácia e na convergência do modelo, especialmente em comparação com métodos mais tradicionais como o SGD.

Também destaca-se que a importância da seleção adequado do *scheduler*, onde nos experimentos que incorporaram *schedulers* de taxa de aprendizado, como OneCycleLR e CosineAnnealingWarmRestarts, mostraram resultados superiores em termos de estabilidade e eficiência durante o treinamento. Isso pode ser atribuído à capacidade desses *schedulers* de ajustar dinamicamente a taxa de aprendizado, evitando oscilações e acelerando a convergência.

Sobre o conjunto de dados utilizado - o Fairface, indica ter uma boa diversidade de imagens, o que é crucial para o treinamento de modelos de reconhecimento facial. No entanto, foi importante garantir que todas as classes estivessem igualmente representadas para evitar viés no desempenho do modelo.

Sobre o algoritmo de reconhecimento facial MTCNN se mostrou eficaz, mesmo com as rotações angular das imagens biométricas.

5.2 Impactos e Contribuições

Este trabalho contribui para a compreensão e aplicação de redes neurais no reconhecimento de faces, evidenciando que a escolha da função de perda e do otimizador desempenha um papel fundamental na obtenção de melhores resultados. Para a indústria,

a adoção dessas técnicas pode significar maior segurança em sistemas de autenticação facial, especialmente em setores financeiros, como na implementação de soluções de identificação de clientes. Para a sociedade, os benefícios incluem a melhoria de tecnologias de reconhecimento em larga escala, garantindo maior precisão e minimizando falhas devido a conjunto de dados desequilibrados e com viés.

5.3 Direções Futuras de Pesquisa

Futuras direções de pesquisa podem explorar a melhoria dos modelos utilizados para lidar com variações de iluminação, pose e expressões faciais, tornando o reconhecimento facial mais robusto em cenários do mundo real. Além disso, pode ser interessante explorar técnicas adicionais de *data augmentation* e *transfer learning* com modelos pré-treinados, utilizando ainda novas arquiteturas avançadas CNNs.

Para uma avaliação mais profunda e melhora das métricas, principalmente a acurácia, pode ser explorado em trabalhos futuros:

- Ativações intermediárias: Para entender como a rede está tomando suas decisões, é possível visualizar as ativações intermediárias das camadas convolucionais, mostrando como diferentes regiões da imagem estão contribuindo para a decisão da rede.
- Análise de Canais de Cor: Durante o treinamento, a rede pode estar aprendendo a distinguir características associadas a tons de pele, como a intensidade de cores em certas regiões, portanto o objetivo é capturar os "padrões de cor"que a rede está aprendendo e verificar se os neurônios específicos estão focando nesses padrões.
- Visualização de Neurônios com técnicas como Grad-CAM: Para identificar quais quadrantes da imagem a rede considera mais relevantes para a classificação de cada raça, é possível utilizar o Grad-CAM (Gradient-weighted Class Activation Mapping).

Por fim, o trabalho e o algoritmo gerados podem ser utilizados em grandes conjunto de dados públicos, para realizar o equilíbrio do conjunto no que tange a raça e equidade, trazendo consigo o tratamento de vieses no reconhecimento facial.

5.4 Conclusão

Através dos experimentos conduzidos e das análises realizadas, este trabalho demonstrou a eficácia das redes neurais convolucionais no reconhecimento de biometria facial. Com a aplicação adequada de técnicas como o CrossEntropyLoss e otimizadores avançados como AdamW, é possível alcançar resultados expressivos. A pesquisa ainda revela os desafios que permanecem, mas também aponta caminhos promissores, como o uso do aprendizado profundo e a evolução dos conjuntos de dados, tornando-os mais equilibrados, mitigando assim os vieses existentes de raças.

REFERÊNCIAS

ABADI, M. *et al.* Tensorflow: A system for large-scale machine learning. *In*: **12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)**. [*S.l.: s.n.*], 2016. p. 265–283.

ACADEMY, D. S. **Deep Learning Book**. Academy, D. S., 2018. Disponível em: https://deeplearningbook.com.br.

AHONEN, T.; HADID, A.; PIETIKÄINEN, M. Face description with local binary patterns: Application to face recognition. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 28, n. 12, p. 2037–2041, 2006.

AZURE, M. **Azure GPU Documentation**. 2024. https://learn.microsoft.com/en-us/azure/virtual-machines/sizes-gpu. Accessed: 2024-09-30.

BATISTA, G. E. A. P. A.; PRATI, R. C.; MONARD, M. C. A study of the behavior of several methods for balancing machine learning training data. **ACM SIGKDD Explorations Newsletter**, Association for Computing Machinery, v. 6, n. 1, p. 20–29, 2004.

BELHUMEUR, P. N.; HESPANHA, J. P.; KRIEGMAN, D. J. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 19, n. 7, p. 711–720, 1997.

BENGIO, Y. **Learning Deep Architectures for AI. Foundations and Trends in Machine Learning, V2(1)**. [*S.l.: s.n.*]: Now Publishers, 2009.

BISHOP, C. M. Training with noise is equivalent to tikhonov regularization. **Neural computation**, MIT Press, v. 7, n. 1, p. 108–116, 1995.

BRADSKI, G. The opencv library. **Dr. Dobb's Journal of Software Tools**, v. 25, n. 11, p. 120–125, 2000.

BUOLAMWINI, J.; GEBRU, T. Gender shades: Intersectional accuracy disparities in commercial gender classification. *In*: FRIEDLER, S. A.; WILSON, C. (ed.). **Proceedings of the 1st Conference on Fairness, Accountability and Transparency**. PMLR, 2018. (Proceedings of Machine Learning Research, v. 81), p. 77–91. Disponível em: https://proceedings.mlr.press/v81/buolamwini18a.html.

CAO, Q. *et al.* Vggface2: A dataset for recognising faces across pose and age. *In*: IEEE. **2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)**. [*S.l.: s.n.*], 2018. p. 67–74.

CHOLLET, F. **Deep Learning with Python**. [*S.l.: s.n.*]: Manning, 2017. ISBN 9781617294433.

CHOLLET, F. *et al.* **Keras: The Python Deep Learning library**. 2015. Accessed: 2024-06-01. Disponível em: https://keras.io.

DENG, J. *et al.* Arcface: Additive angular margin loss for deep face recognition. *In*: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [*S.l.: s.n.*], 2019. p. 4690–4699.

DERTAT, A. Applied deep learning - part 4: Convolutional neural networks. **Towards Data Science**, 2017. Accessed: 2024-06-01. Disponível em: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2.

DIAKOPOULOS, N. Accountability in algorithmic decision making. **Communications of the ACM**, v. 59, p. 56–62, 01 2016.

DROZDOWSKI, P. *et al.* Demographic bias in biometrics: A survey on an emerging challenge. **IEEE Transactions on Technology and Society**, Institute of Electrical and Electronics Engineers (IEEE), v. 1, n. 2, p. 89–103, jun. 2020. ISSN 2637-6415. Disponível em: http://dx.doi.org/10.1109/TTS.2020.2992344.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. **AI Magazine**, v. 17, n. 3, p. 37, Mar. 1996. Disponível em: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1230.

FOUNDATION, P. S. **Python Programming Language**. 2023. Accessed: 2024-06-01. Disponível em: https://www.python.org/.

FRISCHHOLZ, R. W.; DIECKMANN, U. Bioid: A multimodal biometric identification system. **Computer**, IEEE, v. 33, n. 2, p. 64–68, 2000.

FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. **Biological Cybernetics**, v. 36, p. 193–202, 1980.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. *In*: JMLR WORKSHOP AND CONFERENCE PROCEEDINGS. **Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics**. [*S.l.: s.n.*], 2010. p. 249–256.

HACKELING, G. **Mastering Machine Learning with scikit-learn**. Birmingham, UK: Packt Publishing, 2017. ISBN 978-1-78712-836-9.

HAYKIN, S. **Redes Neurais: Princípios e Práticas**. 2. ed. Porto Alegre: Bookman, 2001. ISBN 9788573076103.

HAYKIN, S. **Neural Networks and Learning Machines**. Pearson, 2009. (Pearson International Edition). ISBN 9780131293762. Disponível em: https://books.google.com.br/books?id=KCwWOAAACAAJ.

HE, K. *et al.* Deep residual learning for image recognition. *In*: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [*S.l.: s.n.*], 2016. p. 770–778.

HINTON, G.; SALAKHUTDINOV, R. Reducing the dimensionality of data with neural networks. **Science**, v. 313, n. 5786, p. 504–507, 2006.

HOWARD, A. G. Some improvements on deep convolutional neural network based image classification. **arXiv preprint arXiv:1312.5402**, 2013.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *In*: PMLR. **International conference on machine learning**. [*S.l.: s.n.*], 2015. p. 448–456.

JAFRI, R.; ALI, S.; ARABNIA, H. Face recognition for the visually impaired. *In*: . [*S.l.: s.n.*], 2013.

JAIN, A. K.; ROSS, A.; PRABHAKAR, S. An introduction to biometric recognition. **IEEE Transactions on Circuits and Systems for Video Technology**, IEEE, v. 14, n. 1, p. 4–20, 2004.

KARKKAINEN, K.; JOO, J. Fairface: Face attribute dataset for balanced race, gender, and age for bias measurement and mitigation. *In*: **Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision**. [*S.l.: s.n.*], 2021. p. 1548–1558.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *In*: **Advances in Neural Information Processing Systems**. [*S.l.: s.n.*], 2012. p. 1097–1105.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.

LECUN, Y. *et al.* Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, IEEE, v. 86, n. 11, p. 2278–2324, 1998.

LEIJNEN, S.; VEEN, F. v. The Neural Network Zoo. *In*: **IS4SI 2019 Summit**. MDPI, 2020. p. 9. Disponível em: https://www.mdpi.com/2504-3900/47/1/9.

LI, F.; WU, J.; GAO, R. Transfer learning and fine-tuning convolutional neural networks. **Journal of Artificial Intelligence Research**, v. 67, p. 123–145, 2023.

LI, S. Z.; JAIN, A. K. (ed.). **Handbook of Face Recognition**. 2nd. ed. New York, NY: Springer Science & Business Media, 2011. ISBN 978-0-85729-931-4.

LOSHCHILOV, I.; HUTTER, F. Sgdr: Stochastic gradient descent with warm restarts. *In*: **International Conference on Learning Representations (ICLR)**. [*S.l.: s.n.*], 2017.

LOSHCHILOV, I.; HUTTER, F. Decoupled weight decay regularization. **arXiv preprint arXiv:1711.05101**, 2018.

MARTIN, K. **Ethics of Data and Analytics**. Boca Raton: CRC Press, 2022. ISBN 978-1-03-221731-4 978-1-03-206293-8.

MCCULLOCH, W.; PITTS, W. A logical calculus of ideas immanent in nervous activity. **Bulletin of Mathematical Biophysics**, v. 5, p. 127–147, 1943.

MINSKY, M.; PAPERT, S. **Perceptrons: An Introduction to Computational Geometry**. Cambridge, MA, USA: MIT Press, 1969.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. **Foundations of Machine Learning**. 2. ed. Cambridge, MA: MIT Press, 2018. (Adaptive Computation and Machine Learning). ISBN 978-0-262-03940-6.

OZZETTI, M. **Facial Recognition Models Mitigating Bias**. 2024. https://github.com/marcellozzetti/Facial-Recognition-Models-Mitigating-Bias. Accessed: 2024-09-30.

PARKHI, O. M.; VEDALDI, A.; ZISSERMAN, A. Deep face recognition. *In*: BMVA PRESS. **Proceedings of the British Machine Vision Conference (BMVC)**. [*S.l.: s.n.*], 2015.

PASZKE, A. *et al.* Pytorch: An imperative style, high-performance deep learning library. *In*: **Advances in Neural Information Processing Systems (NeurIPS)**. [*S.l.: s.n.*], 2019. v. 32.

RASCHKA, S.; MIRJALILI, V. **Python Machine Learning**. [*S.l.: s.n.*]: Packt Publishing, 2017.

ROSENBLATT, F. **The perceptron - A perceiving and recognizing automaton**. Ithaca, New York, 1957.

RUSSELL, S. J. *et al.* **Artificial Intelligence: a Modern Approach**. [*S.l.: s.n.*]: Englewood Cliffs: Prentice Hall, 1995. v. 2.

SCHROFF, F.; KALENICHENKO, D.; PHILBIN, J. Facenet: A unified embedding for face recognition and clustering. *In*: IEEE. **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [*S.l.: s.n.*], 2015. p. 815–823.

SERVICES, A. W. **Amazon S3 Documentation**. 2024. https://docs.aws.amazon.com/s3/. Accessed: 2024-09-30.

SIMARD, P. Y.; STEINKRAUS, D.; PLATT, J. C. Best practices for convolutional neural networks applied to visual document analysis. *In*: CITESEER. **ICDAR**. [*S.l.: s.n.*], 2003. v. 3, p. 958–962.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **International Conference on Learning Representations**, 2015.

SMITH, L. N. Cyclical learning rates for training neural networks. *In*: IEEE. **2017 IEEE Winter Conference on Applications of Computer Vision (WACV)**. [*S.l.: s.n.*], 2017. p. 464–472.

STERR, D. Understanding cross-entropy loss. **Towards Data Science**, 2020. Disponível em: https://towardsdatascience.com/understanding-cross-entropy-loss-with-python-6c3bde67e6f4.

SUN, Y.; WANG, X.; TANG, X. Deep learning face representation by joint identification-verification. **arXiv preprint arXiv:1604.02878**, 2016.

TAIGMAN, Y. *et al.* Deepface: Closing the gap to human-level performance in face verification. *In*: IEEE. **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [*S.l.: s.n.*], 2014. p. 1701–1708.

TURK, M.; PENTLAND, A. Eigenfaces for recognition. **Journal of Cognitive Neuroscience**, MIT Press, v. 3, n. 1, p. 71–86, 1991.

WENG, J.; AHUJA, N.; HUANG, T. S. Cresceptron: a self-organizing neural network which grows adaptively. *In*: IEEE. **International Joint Conference on Neural Networks (IJCNN)**. [*S.l.: s.n.*], 1992. v. 1, p. 576–581.

ZHANG, K. *et al.* Joint face detection and alignment using multitask cascaded convolutional networks. **IEEE Signal Processing Letters**, IEEE, v. 23, n. 10, p. 1499–1503, 2016.

ZHANG, X. *et al.* Shufflenet: An extremely efficient convolutional neural network for mobile devices. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2018. p. 6848–6856.

ZHAO, W. *et al.* Face recognition: A literature survey. **ACM computing surveys (CSUR)**, ACM, v. 35, n. 4, p. 399–458, 2003.